Commodore COMPUTER CLUB

29 · L. 3.500

La rivista degli utenti di sistemi Commodore

Mensile - 25 Marzo 1986 - Anno V - N. 29 - Sped. Abb. Post. Gr. III/70 - CR - Distr.: MePe

Speciale linguaggi

Venti domande sul linguaggio macchina

Turbotape sonoro

A caccia di adventures

Come vincere in borsa

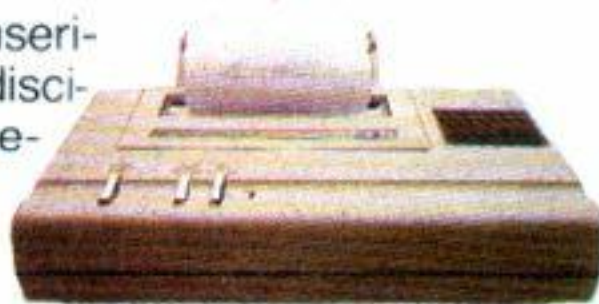
VINCE CHI LEGGE.

Parte l'Operazione Fedeltà Systems. I premi?
Tanti: una moto Cagiva Electra 125, printer-plotter e stampanti a margherita Commodore, abbonamenti alla tua rivista Systems preferita e libri della biblioteca informatica Systems. Partecipare è semplicissimo.

Raccogli i bollini dell'Operazione Fedeltà Systems che troverai

OPERAZIONE FEDELTÀ SYSTEMS.

da oggi fino al 31 marzo 1986 - su Commodore Computer Club, Computer, Commodore, Commodore Club, MSX, Personal Computer, Sinclair Computer, 16/48, Special Systems, VR Videoregistrare e sui libri Systems.

incollali sulla cartolina inserita in ogni rivista e spediscila. Ogni mese parteciperanno all'estrazione tutte le cartoline con almeno 5 punti. E ricorda: l'abbonamento a qualsiasi rivista Systems vale subito 5 punti. Inoltre tutte le cartoline parteciperanno al grande concorso finale: in palio una potente moto Cagiva Electra 125.

REGOLAMENTO DEL CONCORSO

Partecipano all'estrazioni mensili dell'OPERAZIONE FEDELTA' SYSTEMS tutte le cartoline pervenute con almeno 6 punti.
L'abbonamento a una qualsiasi rivista vale 5 punti subito.
I bollini che compaiono su Computer, Commodore Club, 16/48, VR Videoregistrare e i libri Systems valgono 2 punti; quelli di Commodore Computer Club, Commodore e Sinclair Computer 1 punto; quelli delle cassette Special Systems valgono 3 punti.
Il monte premi delle estrazioni mensili è così ripartito:
- dal 1° al 5° estratto: 1 stampante Commodore.
- dal 6° al 10° estratto: 1 printer plotter Commodore.
- dal 11° al 40° estratto: un libro a scelta della biblioteca informatica Systems oppure a scelta un abbonamento ad una rivista Systems.
Tutte le cartoline inviate partecipano all'estrazione finale di una moto Cagiva Electra 125 cc.

EDITORIALE SYSTEMS
"Operazione Fedeltà"
V.le Famagosta, 75
20142 Milano

Sommario

RUBRICHE

Direttore: Alessandro de Simone
Redazione/collaboratori: Claudio Baiocchi, Carlo e Lorenzo Barazzetta, Giovanni Bellù, Simone Bettola, Andrea e Alberto Boriani, Diego e Federico Canetta, Giancarlo Castanga, Pasquale D'Andreti, Maurizio Dell'Abate, Marco De Martino, Piero Dell'Orte, Luca Galluzzi, Michele Maggi, Giancarlo Mariani, Flavio Molinari, Claudio Mueller, Enrico Scelsa, D. Matturro, M.L. Nitti, Massimo Pollutri, Carla Rampi, Fabio Sorgato, Giovanni Verrelli, Antonio Visconti.
Segreteria di redazione: Maura Ceccaroli, Piera Perin
Ufficio Grafico: Mary Benvenuto, Arturo Ciaglia, Paolo Vertuccio
Direzione, redazione, pubblicità: V.le Famagosta, 75 - 20142 Milano - Tel. 02/8467348
Pubblicità: Milano: Leandro Nencioni (direttore vendite), Giorgio Ruffoni, Roberto Sghirinzetti (settore informatica), Claudio Tidone - V.le Famagosta, 75 - 20142 Milano - Tel. 02/8467348
- Emilia Romagna: Spazio E - P.zza Roosevel, 4 - 40123 Bologna - Tel. 051/236979
- Toscana, Marche, Umbria: Mercurio Srl - via Rodari, 9 - San Giovanni Valdarno (Ar) - Tel. 055/947444
- Lazio, Campania: Spazio Nuovo - via P. Foscari 70 - 00139 Roma - Tel. 06/8109679

Segretaria: Lilliana Degiorgi - **Abbonamenti:** Marina Vantini
Tariffe: prezzo per copia L. 3.500. Abbonamento annuo (11 fascicoli) L. 35.000. Estero: il doppio. Abbonamento cumulativo alle riviste Computer e Commodore Computer Club L. 70.000.
I versamenti vanno indirizzati a: Systems Editoriale Srl mediante assegno bancario o utilizzando il c/c postale n. 37952207
Composizioni: Systems Editoriale Srl - **Fotolito:** Systems Editoriale Srl
Stampa: La Litografica S.r.l. - Busto Arsizio (VA)
Registrazione: Tribunale di Milano n. 370 del 2/10/82 - Direttore Responsabile: Michele Di Pisa
Sped. in abb. post. gr. III - Pubblicità inferiore al 70% - **Distrib:** MePe, via G. Carcano 32 - Milano

l'argomento del mese

La lingua batte dove il software duole

Per risolvere un problema col calcolatore è necessario utilizzare il linguaggio più idoneo. Quale usare, tra i tanti? E, soprattutto, che diavolo è un linguaggio?

Si sente tanto parlare di linguaggi, di sistemi operativi, di procedure orientate alla macchina o all'utente e di tante altre belle cose che hanno tutte in comune tra loro (oltre all'intento segreto di far impazzire gli utenti) la possibilità di dialogare con l'utilizzatore del computer.

Troppo spesso, però, non si hanno le idee chiare sui confini esistenti tra sistema operativo e linguaggio; oppure si sente parlare, spesso solo per sentito dire, di differenze modeste o sostanziali tra un linguaggio e l'altro.

Per confondere le idee si asserisce, inoltre, che il Basic, ormai, non ha più motivo di esistere, mentre altri ribattono che il Pascal non è proprio l'optimum per le applicazioni generali.

Poi bisognerebbe parlare di linguaggi interpreti e compilati, di Forth, di "C", di Cobol, di Fortran, di Assembler, di compilatori e decompilatori, di linguaggio macchina e procedure di ingresso uscita che richiederebbero linguaggi e accorgimenti più idonei.

In tutto queto parlare, l'hobbysta "medio" non può che rimanere disorientato, specie se principiante. Chi ha appena, con orgoglio, digitato un mini listato da una riga e ne è soddisfatto, cade nella disperazione più profonda leggendo che il Basic deve abbandonarlo al più presto se vuole restare al passo con i tempi.

Chiariamoci le idee

Sarebbe opportuno ricordare che alla base di tutto c'è solo il microprocessore, potente e veloce aggregato di microcircuiti elettronici talmente miniaturizzati da esser concentrati in pochi millimetri quadrati di silicio.

"Attorno" al micro vengono posizionati al-

Informazioni sicure

Risultati di ricerche elaborati a lungo termine. Idee ponderate e cognizioni dettagliate, di sicura affidabilità e sempre disponibili: queste le Vostre esigenze.

Supporti di informazioni Maxell, gli affidabili. Un risultato di prove rigorosissime e di ricerche pluriennali.

La Vostra scelta più logica per informazioni sicure.

Via M. Civitali 75 · 20148 Milano
Tel.: 02/4047648 · Tx.: 335654

maxell®
supporti magnetici
l'affidabilità

Maxell Europe GmbH · Emanuel-Leutze-Straße 1 · D-4000 Düsseldorf 11 · Tel.: 00 49 / 211 / 59 51 - 0 · Tx.: 8 587 288 mxl d

Lire **10.000**

commodore
COMPUTER CLUB

presenta

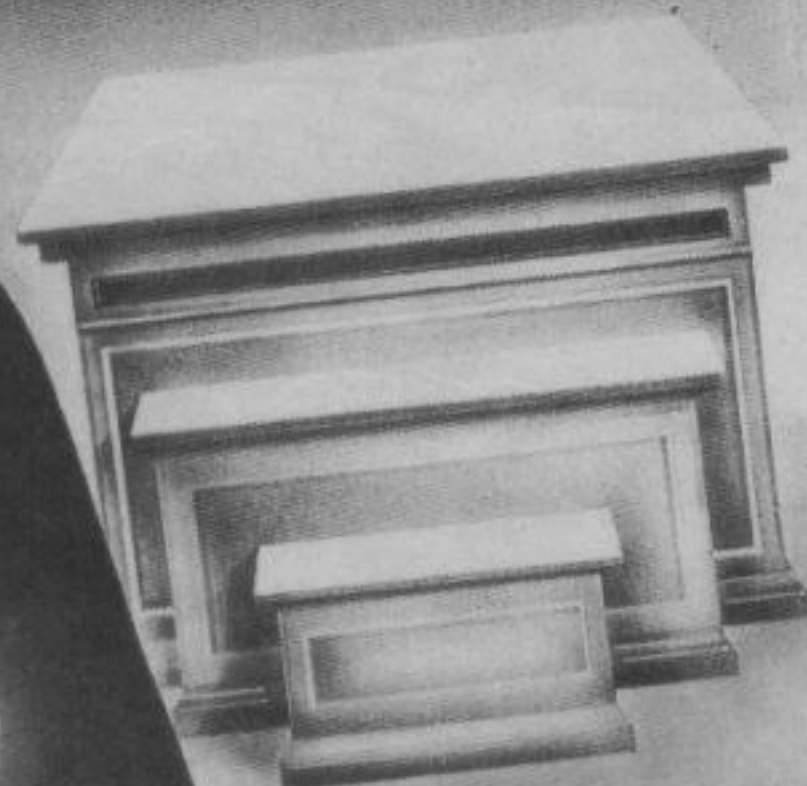
Commodore Club SPECIAL

Suppl. al n 4

Fantastico!

Disegna col solo joystick sullo schermo del tuo 64

RAFFAELLO 64

Richiedilo alla Redazione

tri circuiti integrati in parte indispensabili, in parte "optional".

Il micro, in effetti, non può quasi mai funzionare da solo e richiede per il suo corretto funzionamento altri componenti elettronici. Tra i primi figura il quarzo che, opportunamente eccitato, genera una frequenza che sarà tenuta come riferimento per la cadenza delle operazioni da compiere. Il quarzo rappresenta il cuore che, con i suoi battiti, consente al sistema di vivere.

Alcune elaborazioni compiute dal micro devono esser memorizzate da qualche parte perchè, in seguito, saranno riprese dallo stesso micro ed elaborate ulteriormente.

Facciamo un paragone con una calcolatrice tascabile: se questa ha poche memorie (spesso ne hanno addirittura una sola) il suo utilizzo è notevolmente limitato perchè se, in un gruppo di calcoli concatenati, risulta necessario utilizzare più dati precedenti, è indispensabile scriverli a parte, ad esempio su un foglio di carta.

Con ciò vogliamo solo asserire che la memoria di un calcolatore è indispensabile per un suo corretto funzionamento e, ovviamente, più memoria c'è, più "cose" è possibile fare.

Di memorie, come è noto, esistono diversi tipi, tra cui:

- RAM in cui è possibile scrivere un dato, leggerlo, cancellarlo o modificarlo;
- ROM in cui è solo possibile leggere dati. Questi, naturalmente, sono stati "scritti" in fabbrica e non è possibile modificarli;
- PROM in cui è possibile scrivere dati (mediante un particolare apparecchio chiamato programmatore di Eprom) e successivamente, dopo averla inserita nel calcolatore, sarà possibile leggerli. Volendo, questo tipo di memorie possono esser cancellate e riutilizzate in altro modo.

Continuando nel paragone col corpo umano, come il quarzo rappresentava il cuore, così le ROM (o le EPROM) possono esser considerate la struttura cerebrale comune a tutti gli esseri umani. Le ROM sono, in altre parole, le cellule del cervello in cui sono memorizzate le informazioni relative al camminare, al vedere, gli stimoli della fame, della sete, la possibilità di articolare le braccia, eccetera.

Le memorie RAM, invece, possono esser paragonate alle informazioni che il singolo essere umano acquisisce durante la sua esistenza. Le informazioni ricevute da un uomo durante uno spettacolo televisivo (e gli stimoli che "elabora"), ad esempio, sono completamente diverse da quelle che, nello stesso momento, percepisce un altro essere umano che legge un libro.

I due tipi di informazioni, comunque, possono esser memorizzati grazie alla funzione della vista, alla esistenza di memoria disponibile, al cuore che batte e al cervello che, grazie ad esso, funziona e sovrintende alle varie operazioni di memorizzazione.

Oltre al micro, alle ROM e alle RAM, trovano posto, all'interno del calcolatore, anche i circuiti integrati relativi all'ingresso e all'uscita delle informazioni, detti I/O dall'inglese Input Output.

E' indispensabile, infatti, non solo introdurre informazioni nel sistema, ma anche farle uscire. Tutto questo perchè il sistema sia considerato non solo "intelligente" ma anche funzionante.

Le orecchie, gli occhi, il palato e tutti gli organi preposti all'individuazione dei cinque sensi sono, a tutti gli effetti, gli strumenti di ingresso del nostro corpo.

La parola e i gesti, al contrario, ne rappresentano quelli di uscita.

Senza uno strumento di ingresso, infatti, il sistema non può funzionare pur se perfettamente in ordine. Analogamente un sistema privo di organi di uscita non serve a nulla perchè, tra l'altro, non è possibile stabilire se sta funzionando oppure no.

Chi è cieco dalla nascita può egualmente ricevere, ed elaborare, informazioni grazie agli altri quattro sensi di cui dispone. Allo stesso modo un muto può comunicare con gesti, con carta e penna e, perchè no?, con un terminale computerizzato.

Chi invece, dalla nascita, è privo dei cinque sensi oppure ha la disgrazia di non poter utilizzare gli organi di Output (in seguito, ad esempio, a una paralisi totale) non può elaborare dati, a causa della mancanza di ingressi, oppure non può evidenziare ciò che ha elaborato. Solo sofisticate tecniche di analisi, relative allo studio di impulsi cerebrali, possono stabilire se un essere umano, che si trovi in una situazione così infelice, è da considerarsi vivo oppure no.

Allo stesso modo un calcolatore dotato dei soli organi di ingresso (tastiera, penna ottica, joystick, eccetera) non potrà esser utile se privo di strumenti idonei a evidenziare il risultato delle sue elaborazioni (video, stampante, drive per floppy, plotter e altri apparecchi).

Uno zoom sulla memoria

Che c'entra quello che abbiamo detto con i linguaggi e i sistemi operativi?

La faccenda assume un significato ben preciso se teniamo presente che un byte, vale a dire una cella di memoria, può contenere indifferentemente un dato oppure un'istruzione.

Il numero 4, tanto per fare un esempio banale, può rappresentare un índice (il quarto libro sullo scaffale) oppure un ordine, come il tasto "4" presente in un ascensore che, se premuto, "ordina" di salire al quarto piano di un edificio.

Anche nel linguaggio Basic il numero 4 può assumere diversi significati: numero di linea (4 PRINT"PIPPO"), parte di nome di variabile (A4=150), carattere alfanumerico all'interno di una stringa (PREMI IL TASTO 4 PER CONTINUARE).

Continuando con l'analogia, un gruppo di istruzioni può appartenere, o meno, al programma principale oppure a un sottoprogramma: l'unico modo per stabilirlo consiste nel ricercare il comando Return al termine del gruppo di istruzioni.

Viaggio nel micro

Finalmente arriviamo al nocciolo della questione: qualunque sia il computer utilizzato, qualunque sia il sistema operativo inserito, qualunque sia il linguaggio selezionato, ciò che conduce ai risultati desiderati è sempre e soltanto il microprocessore, le memorie e il sistema di I/O.

Ciò significa che il Basic come il Pascal sono, alla fine dei conti, una serie di istruzioni in Linguaggio Macchina raggruppate in modo tale da accettare dati in ingresso e presentarne altri in uscita.

Considerando il linguaggio Basic, per restare in un campo familiare al nostro lettore, un'istruzione del tipo PRINT"PIPPO" è formata da un numero (153) che rappresenta il comando PRINT, un altro numero che indica il carattere di virgolette (34) e altri numeri che, secondo il codice ASCII, indicano i caratteri alfanumerici (80, 73, 80, 80, 79).

Quando premiamo il tasto Return una parte del calcolatore si incarica di esaminare e di interpretare quella particolare successione di numeri. Il nome PIPPO, insomma, potrebbe venir fuori anche senza ricorrere al Basic ma, magari in modo più scomodo, inserendo con altro sistema una serie di dati all'interno del calcolatore.

Vediamo ora di riassumere il discorso:

- il microprocessore necessita sempre, per funzionare, di istruzioni in linguaggio macchina poste in successione logica e coerente;
- il micro non può in alcun modo sapere se il gruppo di istruzioni che sta eseguendo in un particolare momento fa parte di un linguaggio, di una procedura di I/O oppure del sistema operativo;
- affermare che, in un computer, la memoria è organizzata, ad esempio, in blocchi distinti contenenti il sistema operativo in 32K, il linguaggio interprete in 8K ed il sistema di I/O in 4K, è solo una semplificazione di comodo: non esistono istruzioni (in linguaggio macchina) specifiche per il funzionamento del Basic, del sistema operativo oppure delle I/O;
- in molti casi, proprio grazie alle considerazioni precedenti, è possibile sostituire circuiti integrati ROM, aggiungerli (come nel caso di cartucce da inserire nel retro del computer), modificarli dopo averli ricopiati su memoria RAM e così via.

Concludendo

Il calcolatore deve esser considerato come un fascio vibrante di nervi che, opportunamente stimolati, conducono ai risultati voluti.

Le memorie ROM non devono esser "viste" come blocchi distinti e completamente separati tra loro: niente di male se, disassemblandole, vi accorgete che una parte L.M. dell'interprete Basic utilizza una subroutine L.M. del sistema operativo e viceversa.

Se risulta del tutto normale, alterando opportunamente alcune zone di memoria, aggiungere nuove istruzioni al linguaggio oppure se, addirittura, si può modificare il sistema operativo o far funzionare apparecchi di vostra realizzazione collegati al computer, ciò è dovuto solo al fatto che il modo di funzionare di un calcolatore è basato sul linguaggio specifico del microprocessore in esso impiegato, il Linguaggio Macchina.

Tutto il resto (linguaggi evoluti, sistemi operativi, apparecchi di rice-trasmissione) non sono altro che comodità introdotte dall'uomo per facilitare il colloquio con la macchina sfruttando abilmente le risorse, davvero infinite, del cervello al silicio.

Alessandro de Simone

ATTENZIONE!!!
A tutti i Commodore Computer Club

Molti circoli si sono aperti e molti sono usciti allo "scoperto" dopo il nostro invito ad aprire un Computer Club, apparso sul N. 21 di C.C.C.

Allo scopo di rendere un servizio migliore ai nostri lettori che intendano contattare uno di questi simpatici circoli culturali, i segretari dei circoli stessi sono pregati di compilare il seguente tagliando, o sua fotocopia, e di inviarlo in busta chiusa (affrancata secondo le vigenti tariffe postali) a:

Systems Editoriale
Servizio Notizie Computer Club
Viale Famagosta, 75
Milano

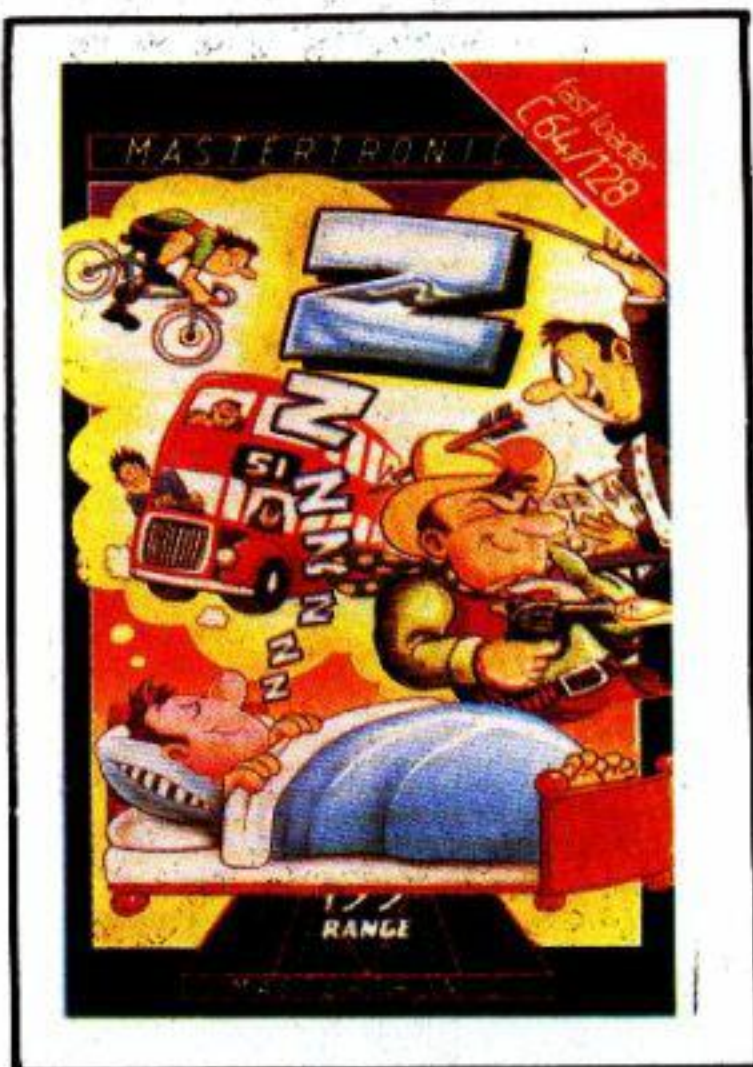
La completa compilazione dell'intera scheda, è INDISPENSABILE per la pubblicazione gratuita sulla nostra rivista.

Nome del club: .... Sede del club: .... Via ....
C.A.P. .... Città .... Prov.: .... Tel. Prefisso: ......N. ....
Presidente: .... Segretario: .... N. soci fondatori: ....
N. di soci finora iscritti: .... Data di fondazione: .... Giorni di apertura della sede: ....
Orario di apertura: .... Computers disponibili (specificare): ....
Periferiche disponibili (specificare): .... Programmi disponibili (N. approssimativo): ....
Videogiochi N.: .... Professionali N.: .... Biblioteca tecnica N. volumi: ....
N. abbonamenti a riviste italiane: .... N. abbonamenti a riviste straniere: .... Quota di iscrizione L. ....
(Specificare se annuale, mensile ecc.) ....
Attività previste: .... Bollettino periodico emesso: .... Attività già svolte: ....
Eventuali sponsor: .... Disponibilità alla sponsorizzazione (sì/no): ....

Il sottoscritto ...., presidente del Computer Club .... autorizzo la Systems Editoriale a diffondere notizie riguardanti le attività del circolo culturale citato anche se pervenute in redazione in via non ufficiale.
Dichiaro inoltre che le informazioni comunicate corrispondono al vero e che, in caso di difformità accertata da parte di incaricati della Systems stessa, Commodore Computer Club, allo scopo di tutelare la buona fede degli utenti della rivista, si riserva il diritto - dovere di avvertire i propri lettori nel modo e nella forma che riterrà più opportuni.

In fede

Il braccio e la mente

Zzzz (C/64 C/128)

Si chiama proprio così (Zzzz) e il motivo è da ricercare nel fatto che si tratta di un adventure "onirico". Si suppone che vi siate addormentati e che nel sogno, dobbiate raggiungere... no, non ve lo diciamo.

Zzzz è un adventure simpaticamente sofisticato; a parte le numerose schermate in alta risoluzione (una per ciascuna "stanza" e, qualcuna, perfino animata), il gioco si avvale della parte destra e sinistra dello schermo per far apparire delle "icone". Queste sono immagini che ricordano le funzioni selezionabili dal giocatore. Posizionandosi col joystick su una delle icone e premendo il pulsante Fire, compaiono, nella parte inferiore dello schermo, i messaggi che consentono al giocatore di effettuare le decisioni più opportune. E' possibile prendere o abbandonare (e vi verrà detto se potete farlo); esaminare (la situazione creatasi e nella quale vi trovate); richiedere una pausa o interrompere il gioco (registrando su nastro il "punto" in cui siete per riprenderlo in seguito); chiedere aiuto (help); ottenere la descrizione di avvenimenti particolari, eccetera.

L'adventure, che si carica in poco più di 132 giri, è scritto in inglese. Questo fatto, però non dovrebbe esser considerato un "difetto" ma, al contrario, un incentivo piacevole per imparare, o perfezionare, una lingua straniera.

E' il caso di dire che si può unire l'utile al dilettevole grazie a un gioco: chi l'avrebbe mai detto...

Collapse (C/64 C/128)

Questo gioco su nastro, che si carica in poco più di 100 giri e che richiede il joystick, può essere definito una specie di Pac-man "evoluto".

Mancano effetti tridimensionali, ma una simpatica animazione supportata da numerosi suoni e rumori e, soprattutto, la notevole varietà di schermate possibili (ben 96!) rendono piuttosto interessante il nuovo videogame proposto dalla Mastertronic.

Descrivere il gioco è piuttosto difficile perchè si tratta di far cambiare di colore, toccandole, delle aste il cui numero e la cui disposizione cambiano notevolmente a seconda della difficoltà impostata.

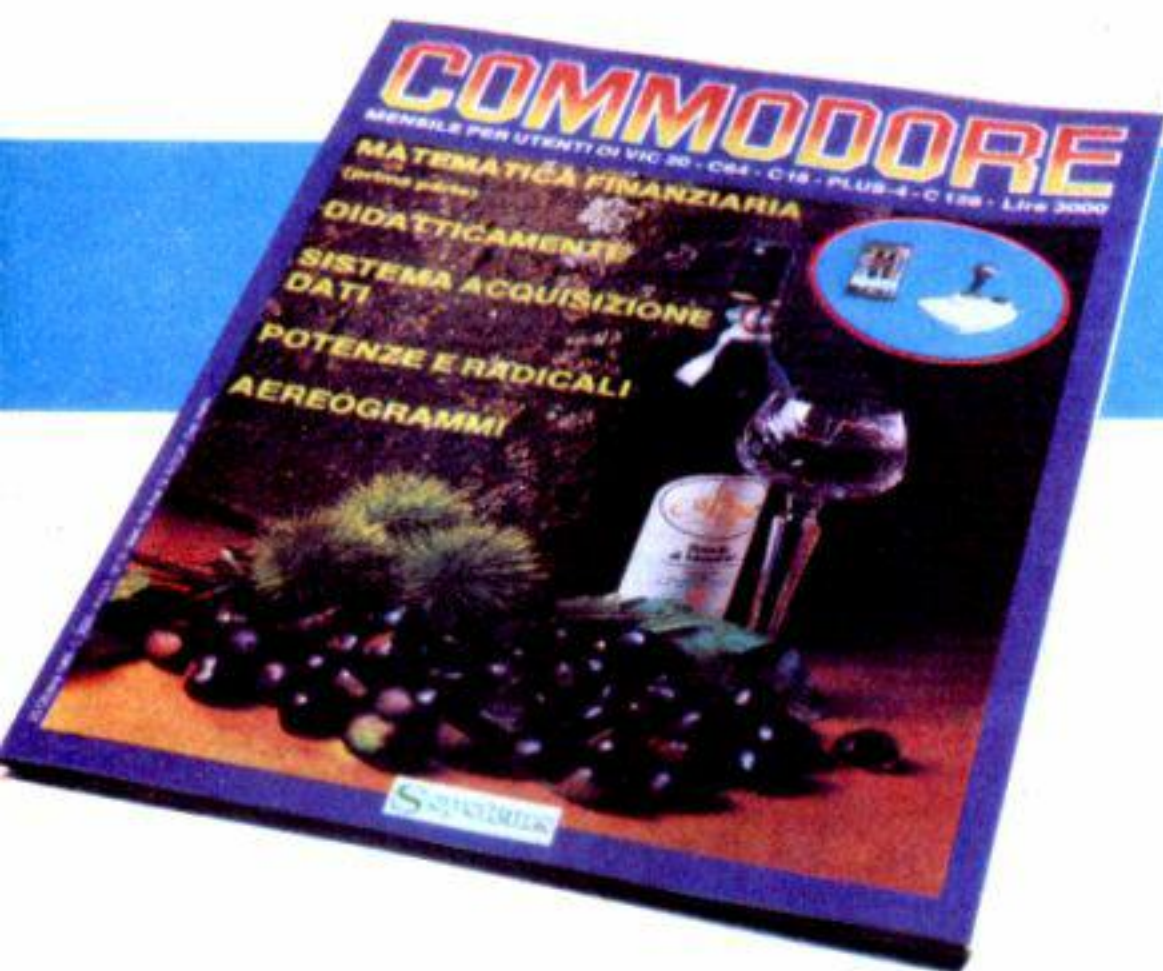
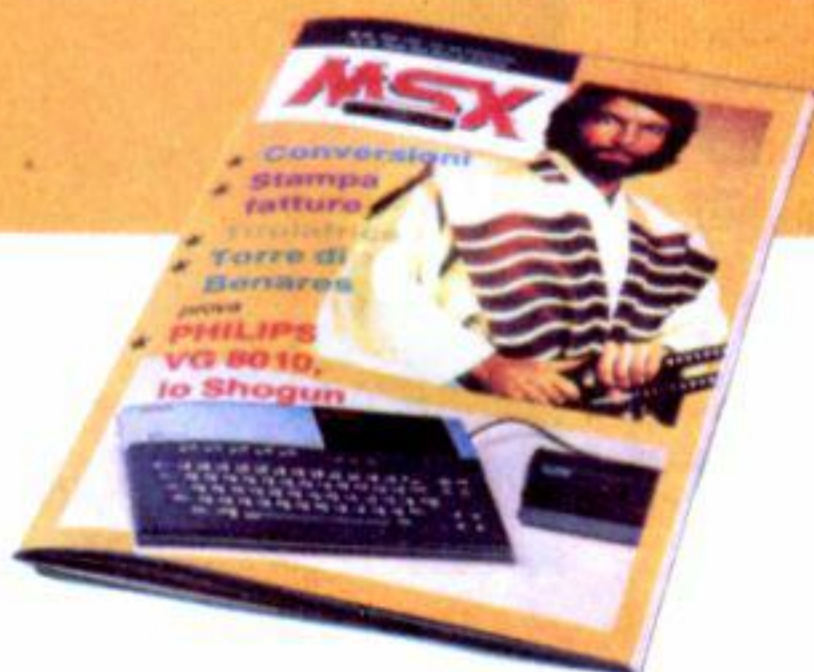
E' necessario non solo cambiare il colore delle aste (in un tempo, tra l'altro, piuttosto contenuto) ma evitare di entrare in collisione con alcune figure che, urtandovi, vi sottraggono, come penalità, tempo prezioso.

Quando, finalmente, tutte le aste sono colorate, è necessario urtarne una in modo tale che questa, ruotando (grazie a una realistica animazione) urti quella adiacente e questa, a sua volta, quella successiva, fino all'ultima.

Sicuramente avrete visto quel gioco che si realizza, appunto sistemando in piedi piccole tavolette di plastica in modo tale che, facendo cadere la prima, si innesti una reazione a catena che porta rapidamente all'abbattimento di tutte le altre.

Molto più complicato di quanto non possa sembrare a prima vista, Collapse è un videogame da non sottovalutare.

128 KBYTES

SINCLAIR COM

COMMODORE

MSX

DI RIVISTA.

PUTER

Personal computer

STUDIO DI FUNZIONE

RILOCATORE DI PROGRAMMI

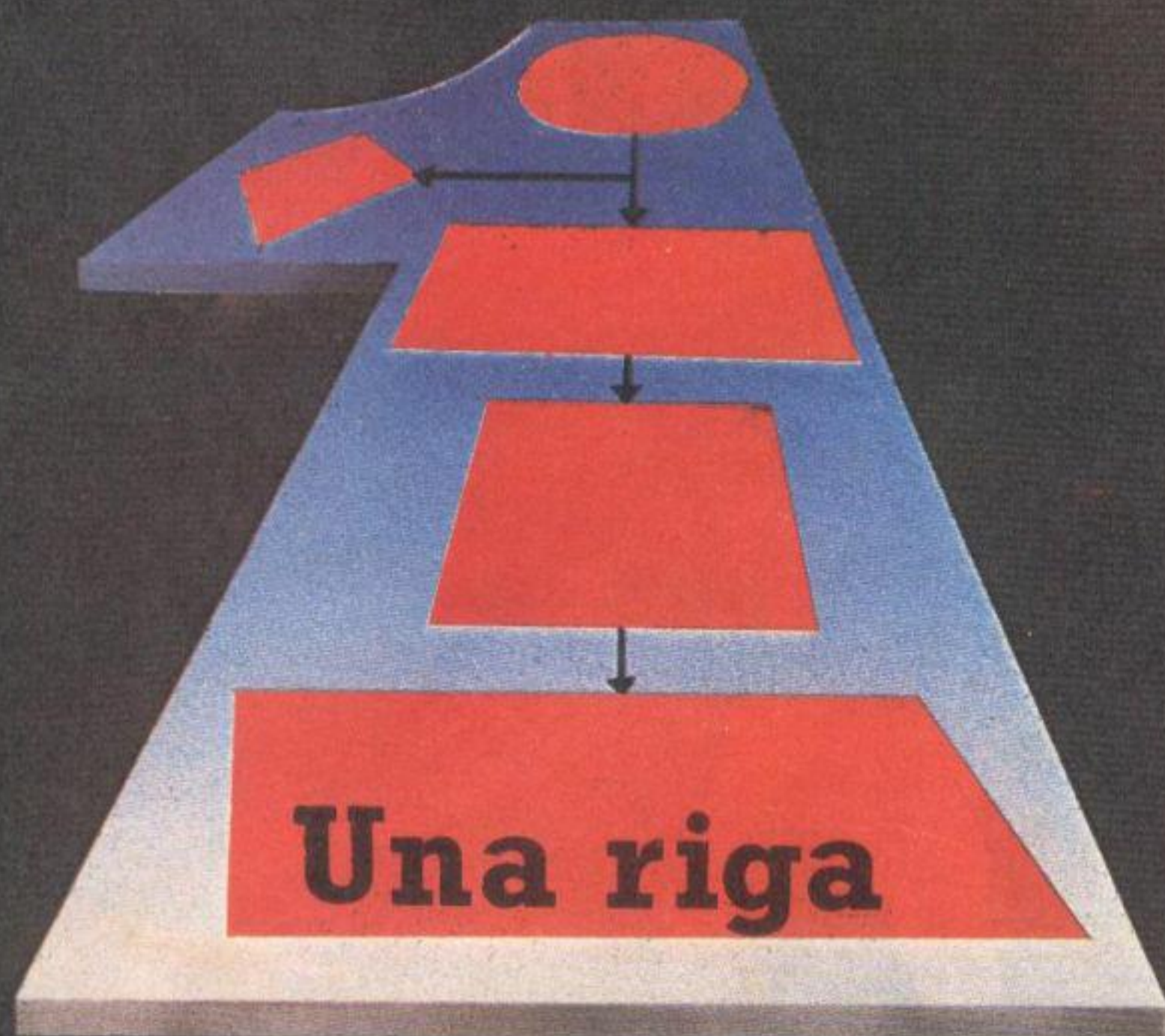
FUNZIONE VAL PER IL QL

TRE RIVISTE IN UNA!

DA GENNAIO £ 3000

Personal Computer è la nuova rivista Systems per gli utenti Commodore, MSX, Sinclair.
Dal mese di gennaio in edicola, 128 pagine a sole 3000 lire: allo stesso prezzo, la vostra rivista, integra e migliorata, e molto di più.
Non solo tre riviste per tre diversi utenti: **Personal Computer** è anche un'idea nuova per far comunicare tutti gli hobbisti.
Personal Computer: 128 Kbytes di rivista, tutti i mesi in edicola.

Il mercato si evolve. Anche noi.

Una riga

Su ogni numero di Commodore Computer Club compaiono un paio di pagine dedicate a una dozzina di micro programmi lunghi una sola riga.

E' ovvio che non è possibile pretendere effetti sorprendenti in listati così brevi, tuttavia, sviluppando l'idea su cui gli stessi programmi son basati è possibile pervenire a realizzazioni di tutto rispetto ricorrendo, è inutile dirlo, a qualche riga in più.

Particolarmente interessanti per i principianti, spesso utili anche per gli esperti, i micro programmi di una sola riga rappresentano una valida "palestra" per abituarsi a concentrare in poche, essenziali istruzioni la soluzione di problemi complessi solo in apparenza.

Anche i mini-listati di questo numero sono opera di Michele Maggi che, stufo del suo fiammante C-128, lo ha recentemente sostituito con un 128-D (quello, per intenderci, col drive 1571 incorporato...).

Per Commodore 64

1

List reset (C-64). Chil'avrebbe mai detto che, con una sola riga, era possibile proteggere i vostri listati? Basterà, infatti, che le istruzioni contenute in questo mini programma siano attivate affinchè un tentativo non autorizzato di List dia come risultato il reset del sistema.

Ai più esperti diremo che viene alterato il vettore di List ponendo, in sua vece, il vettore del System Reset.

Sarà sufficiente dotare i vostri programmi di un auto-run in cui la prima riga di programma contenga le istruzioni pubblicata: chi tenterà di listare i vostri programmi avrà una sgradevole sorpresa!

```
1 POKE 774,226:POKE
   775,252:PRINT"PR
  OVA A FARE IL LIS
  T"
```

2

Restore reset (C-64). Anche questa riga, attivata, altera un vettore. Sarà in seguito sufficiente premere contemporaneamente i tasti Run/Stop e Restore per avere lo stesso effetto dello spegnimento e riaccensione della macchina. Utile per inserire protezioni all'interno dei vostri listati.

```
1 POKE 792,226:POKE
   793,252:PRINT"PR
  OVA A PREMERE RUN
  /STOP E RESTORE..
  ."
```

3

Aspetta tasto (C-64). Per creare una pausa, all'interno di una routine, si ricorre quasi sempre all'istruzione GETA$ che richiede, in genere, un paio di righe. E' perciò possibile, in una sola riga, ricorrere all'istruzione WAIT che obbliga il computer a interrompere l'esecuzione del programma Basic finchè la locazione di memoria indicata dall'"argomento" di Wait non assume un certo valore. E' ovvio che, nel nostro caso, la locazione è quella che viene modificata premendo un tasto qualsiasi.

```
1 PRINT"PREMI UN TA
  STO": POKE 198,0:
  WAIT 198,1
```

4

PRINT At. Digitando questa semplice riga, si posiziona il cursore nel punto indicato dello schermo. La locazione 211 contiene il numero relativo alla colonna (da 0 a 39) e la 214 quello relativo alla riga (da 0 a 24). La SYS "dialoga" direttamente col sistema operativo che posiziona il cursore nella posizione desiderata. Attenti a non superare i valori suggeriti, altrimenti possono accadere cose "strane"...

```
1 INPUT "COL,RIGA";
  C,R:POKE 211,C:PO
  KE 214,R:SYS58640
  :PRINT"ECCOMI IN"
  C;R
```

5

NEW con poke. Un computer Commodore, per sapere se nella sua memoria RAM è allocato un programma, esamina la presenza di tre zeri consecutivi, nell'area destinata all'occupazione di listati Basic. Pokando, appunto, il valore zero proprio nelle prime tre locazioni di tal area, il computer riterrà, erroneamente, di non avere alcun programma al suo interno. Utilizzata in modo intelligente, questa considerazione può tornare utile nell'impostare protezioni di programmi.

```
1 PRINT"PROVA A FAR
  E IL LIST":POKE 2
  048,0:POKE 2049,0
  :POKE 2050,0
```

6

Caratteri colorati. (C-64). Tutti i caratteri del C-64 in tutti i colori possibili. Il tutto in una sola riga.

```
1 FOR I=0 TO 255:PO
  KE 1024+I,I:POKE
  55296+I,I:NEXT
```

7

Scrol fine (C-64). Pochi comandi per iniziare uno studio approfondito sulla gestione dello scrolling del C-64. Eventuali messaggi, insomma, viaggeranno dal basso verso l'alto a "scatti" di un dot (puntino elementare) invece che di una riga interna. Durante il funzionamento, si notano fastidiosi sfarfallii. Ma che cosa si può pretendere con un pugno di istruzioni?

```
1 FOR X=23 TO 16 ST
  EP -1:POKE 53265,
  X:FOR T=1 TO 500:
  NEXTT:NEXTX
```

Per Commodore 16

8

Cerchi (C-16). Potete tentare di ipnotizzarvi nel guardare il disegno che ha origine sfruttando in modo intensivo l'istruzione Circle.

```
1 GRAPHIC1,1:
  FORT=1TO200STEP5:
  CIRCLE1,160,100,T,T
  :NEXT
```

9

Disegno (C-16). Un pianeta? L'orbita di una cometa? Un'anguria tagliata a fette? Fate un po' voi...

```
1 GRAPHIC1,1:
  FORT=1TO200STEP15:
  CIRCLE1,160,100,T,50
  :NEXT
```

10

Typewriter (C-16). Volete un mini-micro-text-editor da usare con la vostra stampante? Cinque istruzioni sono sufficienti. Provare per credere.

```
1 OPEN4,4:GETKEYA$
  :PRINTA$;:
  PRINT#4,A$;
  :RUN
```

11

Caratteri (C-16). Una riga un po' banale, ma preziosissima per chi ha appena comprato il C-16 oppure il Plus/4. Ricordate anche voi i bei tempi lontani quando, per la prima volta, avete sfiorato la tastiera di un computer?

```
1 FORI=0TO255:
  POKE3072+I,I
  :NEXT
```

12

Pop arti (C-16). Gruppi sparsi di rettangoli casuali grazie all'istruzione BOX.

```
1 GRAPHIC1,1:
  FORT=1TO50:
  CIRCLE1,4*T,4*T,
  INT(RND(0)*200),
  INT(RND(0)*100)
  :NEXT
```

13

Pop art2 (C-16). Basta cambiare poche cose, rispetto alla riga precedente, per ottenere un effetto sostanzialmente diverso.

```
1 GRAPHIC1,1:
  FORT=1TO50:
  BOX1,INT(RND(0)*200),
  INT(RND(0)*100),T,3*T
  :NEXT
```

Nota Bene

Alcune righe tra quelle pubblicate sembrano possedere più di 80 caratteri e, come tali, inaccettabili dal computer.

Nei casi in cui ci si accorga che la riga è troppo lunga, è necessario ricorrere alle abbreviazioni dei comandi così come indicato nell'appendice specifica riportata nel manuale del computer in vostro possesso.

Ad esempio invece di scrivere PRINT è possibile abbreviare col punto interrogativo (?). Invece di POKE potete scrivere il carattere "P" seguito dal carattere che viene visualizzato premendo contemporaneamente il tasto shift insieme con "O". Tutte le abbreviazioni possibili, lo ripetiamo, sono riportate in una delle appendici di qualsiasi manuale Commodore.

Nel caso sbagliate a digitare i microlistati che superano, in lunghezza, gli ottanta caratteri (SYNTAX ERROR o altri tipi di errore), è necessario, per sicurezza, ribatterli per intero e non apportare modifiche alla riga visualizzata con l'istruzione LIST.

La partecipazione dei lettori è gradita e compensata, in caso di pubblicazione, con materiale della Systems editoriale (libri, fascicoli arretrati, abbonamenti, eccetera).

Inviate le vostre 1 RIGA, purchè nella misura di almeno dieci per volta. Non ci è infatti possibile raggruppare i (moltissimi) micro-programmi che pervengono singolarmente in Redazione.

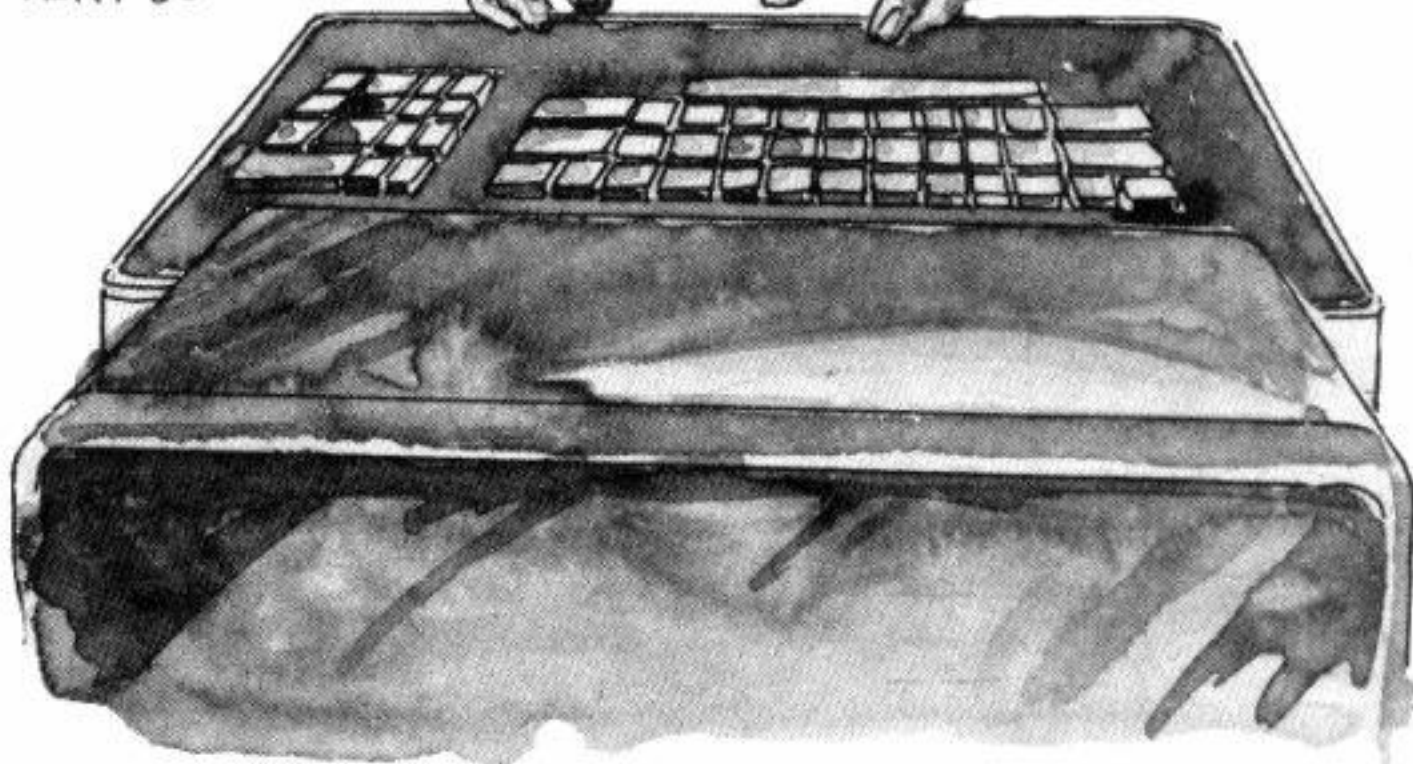
Inviate i vostri lavori su carta, (meglio se su nastro) corredati, ciascuno, di una breve spiegazione sulla funzione che compiono, proprio come li vedete pubblicati in queste pagine.

Ricordate di indicare chiaramente nome, cognome, via e città, e indirizzate a:

Commodore Computer Club
SYSTEMS EDITORIALE

Rubrica "1 Riga"

Viale Famagosta, 75
20142 MILANO

I listati della Systems Editoriale

Un'elevata percentuale dei nostri lettori è alle prime armi nel mondo dell'informatica e incontra difficoltà nel digitare i programmi da noi pubblicati.

I caratteri "speciali" bianchi su fondo nero (semi-grafici in "reverse") che rappresentano precisi comandi per i computer Commodore sono riportati nel listato di esempio a sinistra così come appaiono digitandoli su video o su stampante, mentre a destra come li rappresentiamo nei nostri listati.

La riga 360, ad esempio, deve così esser interpretata:
dopo aver battuto il carattere di virgolette (") che si ottiene premendo il tasto SHIFT insieme con il tasto 2, è necessario battere il carattere CRSR DOWN (il tasto, cioè, che normalmente sposterebbe il cursore nella cella video sottostante).

Analogamente, nella riga 180 del listato "tradotto" (di destra), il termine [NERO] sta a significare che bisogna utilizzare il carattere speciale del colore nero (tasto CTRL insieme con il tasto 1, vedi listato).

Per ricordare in che modo vengono normalmente visualizzati i caratteri speciali, nella seconda parte delle righe di sinistra (dopo i REM) sono riportati i tasti che è necessario premere per ottenere il carattere-comando "speciale".

La Redazione

```
100 REM I CARATTERI SPECIALI
110 REM DEI COMPUTER COMMODORE
120 REM COME APPAIONO NORMALMENTE
130 REM SU VIDEO O SU CARTA.
140 REM (CTRL = TASTO CONTROL)
150 REM (CMDR = TASTO COMMODORE)
160 REM (CRSR = TASTI CURSORE)
170 :
180 PRINT"■":REM CTRL+1 NERO
190 PRINT"■":REM  "  +2 BIANCO
200 PRINT"■":REM  "  +3 ROSSO
210 PRINT"■":REM  "  +4 AZZURRO
220 PRINT"■":REM  "  +5 PORPORA
230 PRINT"■":REM  "  +6 VERDE
240 PRINT"■":REM  "  +7 BLU
250 PRINT"■":REM  "  +8 GIALLO
260 PRINT"■":REM  "  +9 REVERSE ON
270 PRINT"■":REM  "  +0 REVERSE OFF
280 PRINT"■":REM CMDR+1 ARANCIO
290 PRINT"■":REM  "  +2 MARRONE
300 PRINT"■":REM  "  +3 ROSSO CHIARO
310 PRINT"■":REM  "  +4 GRIGIO 1
320 PRINT"■":REM  "  +5 GRIGIO 2
330 PRINT"■":REM  "  +6 VERDE CHIARO
340 PRINT"■":REM  "  +7 BLU CHIARO
350 PRINT"■":REM  "  +8 GRIGIO 3
360 PRINT"■":REM  CRSR IN BASSO
370 PRINT"■":REM  CRSR A DESTRA
380 PRINT"■":REM  CRSR IN ALTO
390 PRINT"■":REM  CRSR SINISTRA
400 PRINT"■":REM  HOME
410 PRINT"■":REM  CANCELLA SCHERMO
420 :
430 REM  ESEMPI DI VISUALIZZAZIONE:
440 PRINT"■■■■■■":REM CANCELLA SCHERMO,
450 :                 REM CRSR DWN DUE VOLTE
460 :                 REM CRSR DESTRA TRE "
470 :
480 PRINT"■■■■":REM BIANCO,CRSR SINISTRA
490 :             :REM DUE VOLTE E CRSR DWN
500 :             :REM UNA SOLA VOLTA
```

```
100 REM  I CARATTERI
110 REM  SPECIALI: COME
120 REM  VENGONO INDICATI
130 REM  SULLE RIVISTE:
140 REM    COMMODORE
150 REM  E COMMODORE
160 REM  COMPUTER CLUB.
170 :
180 PRINT"[NERO]"
190 PRINT"[BIANCO]"
200 PRINT"[ROSSO]"
210 PRINT"[AZZUR]"
220 PRINT"[VIOLA]"
230 PRINT"[VERDE]"
240 PRINT"[BLEU]"
250 PRINT"[GIALLO]"
260 PRINT"[RVS]"
270 PRINT"[RVOFF]"
280 PRINT"[ARANC]"
290 PRINT"[MARR]"
300 PRINT"[ROSA]"
310 PRINT"[GRIGIO1]"
320 PRINT"[GRIGIO2]"
330 PRINT"[VERDE2]"
340 PRINT"[CELESTE]"
350 PRINT"[GRIGIO3]"
360 PRINT"[DOWN]"
370 PRINT"[RIGHT]"
380 PRINT"[UP]"
390 PRINT"[LEFT]"
400 PRINT"[HOME]"
410 PRINT"[CLEAR]"
420 :
430 REM   ESEMPI
440 PRINT"[CLEAR][2 DOWN]
    [4 RIGHT]"
450 :
460 :
470 :
480 PRINT"[BIANCO][2 LEFT
    ][DOWN]"
```

Speciale linguaggi

Sistema operativo, chi era costui?

Pc-Dos, CP/M-80, MS-DOS, PCOS, chi non ha sentito almeno una volta uno di questi nomi? Sono tutti sotto lo stesso denominatore: sistema operativo, SO per gli affezionati alle sigle.

di M.L.Nitti - D.R.Matturro

Parlando di C-64 o Vic-20, spunta qua e là il termine Kernal. A volte si accenna alle routine di sistema (quale sistema?), ma, da quando è venuto alla luce il Commodore 128, le sigle di cui sopra e la definizione "sistema operativo" sono diventati termini comuni anche se, come spesso accade, non tutti sanno che....

Ci siamo posti, quindi, il problema di fare luce su questo "mistero" per accontentare quelli che "non sanno" e quelli che sanno poco, visto che il sistema operativo concorre in misura fondamentale alla definizione delle caratteristiche di un computer.

Una definizione

Buona parte delle attività di un elaboratore consiste nel trasferimento di informazioni tra il microprocessore (cervello della macchina) e le periferiche (drive, stampanti, registratori, plotter, eccetera) e nel controllo di queste ultime.

Per espletare tali funzioni non bastano le parti fisiche, definite anche col termine hardware, ma sono necessari programmi particolari scritti nel linguaggio specifico del microprocessore, che poco ha a che fare col Basic.

Possiamo definire sistema operativo l'insieme di questi programmi.

Per esempio, quando chiediamo al calcolatore di espletare una funzione come quella di Loading (caricamento di un programma), utilizziamo il comando LOAD del Basic. Questo, a sua volta (e senza che ce ne rendiamo conto) esegue una routine dell'interprete Basic che provvede, appunto, al caricamento utilizzando a sua volta le routine del sistema operativo che sovraintendono all'intera operazione. E' un po' come il gioco del telefono in cui i numerosi partecipanti devono dirsi in un orecchio, l'un l'altro, a turno, una parola. Nel caso del computer, ovviamente, i risultati sono più affidabili.

Il comando LOAD, quindi, corrisponde a una routine (programma) in linguaggio macchina che utilizza alcune delle routine (ancora programmi) che costituiscono il sistema operativo.

Da questo gioco di parole risulta che lo stesso interprete Basic dipende dal SO. Questo deve quindi essere scritto in funzione del sistema sul quale dovrà "girare".

Dimmi che SO usi e ti dirò chi sei

A questo punto risulta chiara l'importanza del sistema operativo: sovraintende all'uso dell'hardware e quindi dalla sua "bontà" dipende l'esaltazione delle caratteristiche del microprocessore: da esso dipendono anche i linguaggi di programmazione e da questi ultimi la qualità delle applicazioni.

E' inutile installare un ottimo motore in una carrozzeria inadeguata e tale, comunque, da non sfruttare appieno le doti del propulsore. Viceversa, da un motore di modeste prestazioni è possibile ricavare un'ottima autovettura, se si studiano nei minimi particolari tutti gli accorgimenti che esaltano i pregi e... ne minimizzano i difetti.

Il successo commerciale di un sistema operativo dipende da una catena di particolari che si fondano in parte sulla qualità del chip (microprocessore) e, soprattutto, sull'uso che ne fa il SO.

Software di base

Il sistema operativo è quindi un insieme di procedure così come lo è l'interprete Basic: entrambi appartengono a una categoria di programmi detti software di base. Questi possono essere definiti come gli accessori necessari al funzionamento e all'adeguato sfruttamento delle risorse del microprocessore.

Negli home computer (C-64, Vic-20) questi software risiedono permanentemente nella memoria, anzi in una particolare memoria installata dal fabbricante detta ROM (read only memory: memoria di sola lettura). Questa sigla, in termini discorsivi, può esser tradotta come: una memoria dove non è possibile "pokare". Il software di base, scritto permanentemente su circuito integrato, prende anche il nome di firmware.

I personal computer, a differenza degli home, possiedono prestazioni più elevate (quali maggiori velocità di lettura/scrittura dalle memorie di massa). In

questo caso il sistema operativo non risiede permanentemente su ROM, ma deve essere caricato al momento dell'accensione. In queste macchine gli unici programmi su firmware sono due ed entrambi piuttosto brevi: uno di diagnostica e uno caricatore, detto anche di bootstrap. Il primo controlla l'efficienza delle parti fisiche della macchina prima di dare il run al secondo che provvede al caricamento del sistema operativo.

Una giungla di nomi

A questo punto entrano in gioco le sigle di cui abbiamo parlato in apertura di questo articolo. Quei nomi corrispondono ai sistemi operativi più famosi e tra questi spiccano maggiormente l'MS/DOS (MicroSoft Disk Operative System) della notissima Microsoft e CP/M-80 (Control Process for Microprocessors) dell'altrettanto nota Digital. Il primo è legato al Pc/Ibm (viene infatti chiamato più comunemente Pc/dos) e il secondo è più vicino a noi perchè utilizzabile con il nuovo nato Commodore 128.

Fare un confronto vero e proprio tra i due, però, non sarebbe ragionevole poichè sovraintendono al funzionamento di microprocessori completamente diversi tra loro. Infatti il CP/M-80 è stato fatto su misura per i microprocessori (vecchi ma ancora validi e potenti) Intel 8080 e Z/80. Il sistema MS/DOS sfrutta invece le migliori prestazioni degli Intel 8086-8088 di più recente realizzazione.

Ci limiteremo quindi a illustrare qualche caratteristica dei due sistemi facendo particolarmente riferimento al CP/M-80 che interessa più da vicino i nostri lettori.

Il linguaggio di sistema

Una volta "bootstrappato" (caricato al momento dell'accensione) il sistema operativo, ci troviamo in quello che comunemente si definisce "ambiente" di sistema. A differenza dei sistemi operativi in ROM, come quello del C-64, che sfruttano il Basic come linguaggio di comunicazione con l'utente, qui abbiamo a che fare con una sorta di linguaggio del sistema.

Più che di "linguaggio" dovremmo però parlare semplicemente di un gruppo di comandi. Questi sono soprattutto quelli di gestione delle periferiche e in particolar modo dei disk-drive. Ci riferiamo quindi alla lettura della directory (elenco dei file contenuti in un disco), alla formattazione dei dischi, alla stampa di file/dati, alla copia di dischetti eccetera. Quando il lettore lavorerà su sistemi più "grossi", si renderà conto di persona che la parte relativa al colloquio con le memorie di massa (dischi rigidi, floppy) è la più importante e delicata nel normale lavoro professionale.

Comandi residenti e transienti

In ambiente MS/DOS tali comandi devono essere sempre caricati in memoria prima di essere eseguiti, quindi il disco di sistema deve essere sempre in linea, cioè nel drive. Naturalmente i tempi di caricamento sono ultraveloci, neanche paragonabili a quelli del C-64. E' chiaro che usare tale sistema operativo in presenza di un unico drive è problematico e infatti macchine come il Commodore Pc-10 sono dotate quasi sempre di due drive.
Per ovviare a questo problema il CP/M-80, che funziona spesso su sistemi a un solo drive (come il C-128), possiede alcuni comandi, quali "DIR" per la lettura della directory o "ERA" per la cancellazione dei file e altri ancora, sempre residenti in ROM.

Se qualche utente di C-128 possiede due drive, tanto meglio per lui. Quelli, e sono tanti, che ne hanno uno solo, possono tirare un lungo respiro e dormire sonni tranquilli.

I comandi del CP/M che non risiedono in memoria vengono comunemente detti transienti proprio perchè, una volta utilizzati, non servono più e possono esser cancellati in modo da non occupare più spazi di memoria. Devono, ovviamente, essere caricati di volta in volta per essere eseguiti. Tanto per citarne uno tra i più importanti, possiamo prendere in considerazione PIP (Peripheral Interchange Program) che provvede alla copiatura di dischetti, al trasferimento dati da dischetto a stampante, eccetera.

I software di utilità

Ogni sistema operativo che si rispetti possiede a corredo vari programmi di utilità, che risiedono generalmente accanto ai comandi nel disco di sistema. Sia MS/DOS che CP/M-80 sono dotati di un assemblatore che permette di scrivere programmi nel linguaggio assembler relativo al microprocessore. Un editore di testi permette di leggere e manipolare i file/programma oppure i file dati. Sia chiaro che gli editori di testi in questione trovano grande utilità nel lavoro sui file, ma non sono affatto paragonabili a programmi specifici di wordprocessing.

Come scrivere un programma con i comandi di sistema

pesso si rendono necessarie procedure ripetitive nell'ambiente di sistema e sia in CP/M che in MS/DOS è possibile creare file programma che svolgono procedure di sistema automaticamente, senza la necessaria riscrittura dei comandi. Tale prerogativa è ottenibile con due metodologie differenti:

- il primo consiste nella possibilità di creare, utilizzando appropriate procedure, un file di istruzioni detto "file batch";
- la seconda possibilità consiste invece nell'utilizzo di apposite chiamate (CALL) dal Basic.

Questa seconda ipotesi potrebbe essere paragonata all'utilizzo delle SYS nel C-64, ma solo da un punto di vista teorico: nella pratica queste CALL sono estremamente semplici da usare anche da coloro che non conoscono a fondo il SO e non hanno nessuna nozione di codice macchina.

In definitiva l'approccio a un sistema operativo come CP/M ci introduce in quello che possiamo definire l'ambiente professionale dell'elaborazione dati, aprendoci infinite possibilità di sperimentazione e, perchè no?, di grande divertimento.

Molti sono, infatti, i programmi validi sinora realizzati sul sistema CP/M.

Ora che anche la Commodore è entrata in questo sistema ne vedremo delle belle...

Il Pascal

di M.L. Nitti - D.R. Matturro

Il Pascal, sviluppato da Niklaus Wirth intorno al 1966, è il primo linguaggio coerente con i principi della programmazione strutturata.

Un programma in Pascal si presenta come un corpo iniziale di dichiarazioni, seguito da una serie di procedure che eseguono tutti i lavori di routine.

Il lavoro di stesura del programma deve essere organizzato a tavolino, almeno nella sua struttura globale. Risulta difficile scrivere un buon programma "improvvisato" in Pascal, così come si è abituati in ambiente Basic.

Esistono infatti regole molto rigide di sintassi che obbligano, in un certo senso, a ottimizzare un programma. Tutte le variabili utilizzate, per esempio, devono essere dichiarate in testa al programma e nelle singole procedure. Ciò costringe a tenerle a mente e, di conseguenza, obbliga il programmatore non solo a una "economia" nell'uso, ma anche a essere ordinati e precisi (il che non guasta).

Non esiste numerazione di linee, come nel Basic, anche se è prevista la possibilità di dichiarare "labels" per punti strategici del programma. Con le possibilità offerte, comunque, si riesce a scrivere una procedura che effettui parecchie scelte logiche in modo compatto, senza "salti" di alcun tipo. Nei casi piu difficili è comunque previsto l'uso del famigerato "goto", elemento di disturbo ai fini della logica comprensione di un qualsiasi listato.

Ricorriamo a un paragone, valido soprattutto per i nostri lettori già esperti di Basic.

Pascal:

```
procedure conto
    begin
    volte:=0;
    while (a>b)
        begin
        a:=a-b;
        volte:=volte+1;
        end;
    end;
```

Basic:

```
 990 REM SOTTOPROGRAMMA
1000 I=0
1010 I=I+1
1020 A=A-B
1030 IF A>B THEN 1010
```

Si vuole contare quante volte B è contenuto in A. L'esempio è estremamente semplice, ma già chiarisce alcune differenze. La struttura "while" permette di identificare molto chiaramente quale è la serie di istruzioni che viene ripetuta, e quale è la condizione che deve verificarsi perchè il procedimento continui. Al contrario, in Basic, bisogna osservare molto bene un listato (ovviamente, piu complesso del nostro esempio!) per capire cosa si sta facendo e quale è il punto di rientro.

Le variabili Basic vengono sempre modificate dal programma (nell'esempio varia l'indice "I") cioè sono tutte "globali". In Pascal, invece, le variabili possono essere globali o anche "locali" all'interno della procedura nella quale sono dichiarate. Menzioniamo ancora una differenza: in Pascal esiste un solo punto di chiamata per ogni procedura, mentre in Basic si possono creare situazioni ambigue del tipo:

```
900 GOSUB 1000
...............................................
990 REM SOTTOPROGRAMMA
1000 I=0
...............................................
1040 RETURN
```

in cui il sottoprogramma 1000 è richiamato la prima volta con l'istruzione GOSUB di riga 900, mentre la seconda volta, col "proseguire" del programma, viene eseguito in modo sequenziale, a meno che non si inserisca, ad esempio, 999 END.

L'importanza del Pascal non sta comunque solo in questi miglioramenti di possibilità già esistenti, ma, piuttosto, negli strumenti nuovi che mette a disposizione, tra cui vogliamo ricordare in modo particolare: type, record, pointer.

Oltre ai tipi standard di variabili intere, reali, logiche, caratteri, esiste la possibilità di creare tipi piu astratti, a cui sono applicabili gli operatori di relazione:

```
type giorno = (lun,mar,mer,
gio,ven,sab);

var oggi: giorno;
```

La variabile "oggi" è di tipo "giorno" e può essere confrontata con altre dello stesso tipo; valgono cioè relazioni quali:

```
lun < mar
mar < ven
```

Possono inoltre essere definiti "set" come collezioni di oggetti dello stesso tipo:

```
type periodo = set of giorno;
```

con cui si possono effettuare operazioni di unione, intersezione, differenza di insiemi.

Un "record" è una variabile strutturata con parecchie componenti, che possono essere di tipo diverso:

```
type data = record
            giorno: 1..31;
            mese: 1..12;
            anno:1000..2000;

type persona = record
               nome: string;
               cognome: string;
               nascita: data;
               sesso:(maschile,
               femminile);
               stato:(non sposato,
               spos., vedovo, divorz.);
               var studente: persona;
```

Gli esempi si potrebbero moltiplicare e complicare, ma non è questa la sede adatta. Si noti solamente che l'uso dei record permette una manipolazione dei dati quasi impossibile con altri linguaggi.

L'ulteriore innovazione del Pascal, ora presente anche in altri linguaggi tipo "C", è il "pointer" (puntatore), che permette l'allocazione dinamica dei dati nella fase di esecuzione. Normalmente le dichiarazioni dell'ampiezza di un vettore o di una matrice, vengono fatte all'inizio di un programma e devono essere rispettate.

Se in Basic viene definito un vettore con l'istruzione DIM N(100), nel momento in cui, in fase di esecuzione si tenterà di usare l'elemento N(101), si avrà una segnalazione di errore. Il problema opposto è che se del vettore vengono usati solo i primi dieci elementi, ci sarà una quantità di spazio sprecato.

Ciò che permette il Pascal con l'uso dei puntatori è la creazione dello spazio necessario per i dati solo al momento in cui serve e la restituzione dello stesso quando non serve più.

Con la dichiarazione:

```
type link = persona;
var p: link;
```

si dice che il tipo "link" è un puntatore all'oggetto "persona" e che la variabile "p" è di tipo link. Una variabile di tipo "persona" potrà essere memorizzata creando un puntatore con l'istruzione new(p), mentre sarà cancellata restituendo il suo puntatore con dispose(p). In pratica, il puntatore contiene l'indirizzo della variabile a cui punta.

Si potrebbe paragonare l'uso dei puntatori alla gestione di un vettore fatta direttamente da programma con l'uso delle istruzioni POKE, PEEK, per cui non si avrebbe alcun dimensionamento ma solo un contatore del numero di dati inseriti a partire da un certo indirizzo.

L'argomento è molto complesso e non si può esaurire in questa sede. Appare comunque evidente come la possibilità di creare dinamicamente arrays e record apra la strada a elaborazioni molto complesse, realizzabili in modo economico. Si pensi semplicemente alla gestione di tutti i dati relativi all'archivio di una biblioteca.

Anche il Pascal ha comunque qualche pecca, ma, del resto, nessun linguaggio può offrire il massimo sotto ogni aspetto.

La trattazione di input e output non è particolarmente facile nè comoda, sia per quanto riguarda gli I/O standard, sia per ciò che concerne la manipolazione di file. Salvo provvedimenti particolari presi nelle diverse implementazioni, è quasi impossibile all'utente creare una sua biblioteca di programmi.

Ricordiamo inoltre che il Pascal non è interpretato, ma compilato, ed è quindi uno strumento piuttosto complesso per chi si avvicina per la prima volta a un linguaggio di programmazione.

La chiarezza della strutturazione e la comodità degli strumenti offerti passano però in primo piano rispetto allo svantaggio accennato, superato solo, per il momento, dal Pascal interattivo realizzato per Macintosh.

Il Fortran

Il linguaggio Fortran (Formula Translation) è uno dei più "vecchi" linguaggi ad alto livello, nato assieme al Basic e al Cobol intorno agli anni cinquanta.

di M.L. Nitti - D.R. Matturro

Mentre il Basic era esplicitamente orientato ai principianti e il Cobol alle elaborazioni commerciali, si voleva fare del Fortran uno strumento per le elaborazioni di tipo scientifico.

L'idea base era che un programmatore avrebbe dovuto scrivere i programmi che gli interessavano in un linguaggio il più possibile familiare.

Questo strumento avrebbe dovuto permettere una facile "traduzione" di formule matematico-scientifiche, per farle poi elaborare al calcolatore.

Ancora oggi il Fortran rimane il linguaggio più largamente usato per lavori scientifici e tecnici.

Per un certo periodo ha subito numerose "trasformazioni" per cui si resero disponibili "dialetti" praticamente incompatibili tra loro: un programma scritto per una macchina non poteva "girare" su un'altra senza fare molteplici modifiche.

A partire dal 1966 l'American National Standards Institute (Ansi) ha definito una forma standard di Fortran, chiamata appunto Ansi Fortran, con lo scopo di renderlo "portabile", ovvero di fare in modo che un qualsiasi programma potesse girare su macchine, anche diverse, pur se con lievissimi ritocchi.

Resta comunque un problema la molteplicità delle versioni, anche se standard, esistenti: FortranIV, Fortran66, Fortran77, Fortran77+, per non citare il vecchissimo FortranII, per cui un eventuale utente può restare totalmente disorientato. Del resto questo è un problema comune ai linguaggi più vecchi. Chiaramente non potrà sorgere per linguaggi relativamente giovani, come il "C" e il Pascal.

Parliamo ora delle caratteristiche del Fortran, riferendoci alla sua versione più "bella", il Fortran77+.

In dettaglio

Come si presenta un programma scritto in Fortran?

Esiste un blocco iniziale di dichiarazioni, contenente variabili, dimensionamenti vari, eventuali variabili globali, eventuale inclusione di file. Segue un

programma principale e una lista di subroutine.

Le variabili intere e reali non vanno necessariamente dichiarate, mentre quelle logiche e le variabili stringa (characters) devono essere contenute in apposite dichiarazioni. Uno degli svantaggi di questo linguaggio è che le variabili, per essere globali (valide per l'intero programma), vanno dichiarate in ogni singola routine, oltre che nel "main program" (programma principale). Un vantaggio è invece rappresentato dal fatto che in routine diverse possono tranquillamente trovarsi variabili con nomi identici, senza creare problemi.

La versione citata del Fortran contiene tutte le frasi di controllo che appartengono ai linguaggi più evoluti, tra cui i salti condizionati (if then else), cicli con condizioni imposte all'inizio della dichiarazione (do while, do until), scelte tra vari casi possibili (select case). Permette, quindi, di realizzare una programmazione strutturata di buon livello, rendendo inutili, di fatto, etichette e "goto". Le etichette sono comunque permesse nel programma.

Chiariamo per esempio, per i nostri lettori più esperti di Basic, il significato del "do while". Mettiamo a fronte questi due spezzoni di programma:

Basic

```
 990 I=I+1
1000 GET A$:IF A$=""THEN 1000
1010 IF A$="S" THEN 990
1020 PRINT 'FINE'
1030 END
```

Fortran

```
a = 's'
do while (a.eq.'s')
i=i+1
read*,a
end do
print*,'fine'
end
```

In Basic esiste la possibilità di creare un ciclo solamente su valori numerici; per realizzare quindi un ciclo basato su una condizione vera o falsa, bisogna ricorrere a un "if" unito al salto.

In Fortran il ciclo "do", unito con "while", prosegue finchè è vera la condizione riportata in testa.

Vediamo ora come si presenta un piccolo programma completo, che calcola la somma di cento elementi letti uno per volta e capace di individuare il valore minimo, il massimo e il range degli elementi:

```
C     TROVA LA SOMMA, MINIMO, MASSIMO, RANGE
      REAL MAX, MIN
      READ *,X
      SOMMA = X
      MAX = X
      MIN = X
      DO I=2,100
         READ *, X
         IF (X.GT.MAX) THEN
            MAX = X
         ELSEIF (X.LT.SMALL) THEN
               MIN = X
         ENDIF
         SOMMA = SOMMA + X
      END DO
      RANGE = MAX - MIN
      PRINT *, 'SOMMA = ',SOMMA,'MASSIMO =',MAX,
      1'MINIMO = ',MIN,'RANGE =',RANGE
      END
```

Purtroppo non esiste la possibilità di allocazione dinamica della memoria: le dimensioni vanno fissate all'inizio in modo rigido. Inoltre questo linguaggio non permette in alcun modo di programmare a un livello più basso: non esiste possibilità alcuna di leggere o modificare, da programma, il contenuto di indirizzi di memoria specifici.

Il Fortran si presta molto bene per uso didattico. E' certo meno facile da digerire di quanto non lo sia il Basic, ma al suo confronto permette un tipo di programmazione molto più raffinata. Una delle maggiori difficoltà per i principianti è sicuramente costituita dal problema della compilazione, che aumenta ovviamente i tempi tra la scrittura del programma e l'osservazione dei risultati cui porta.

Anche la possibilità di manipolazione dei caratteri è molto buona: esistono funzioni per contare la lunghezza delle stringhe, per accostarle, per isolare i singoli caratteri.

Purtroppo sono ancora in circolazione versioni meno dotate, come il FortranIV, che non possiedono l'istruzione "if then else" e neppure le funzioni di manipolazione delle stringhe.

La gestione dei file risulta semplice, forse tra le più semplici per ciò che concerne i linguaggi ad alto livello. La flessibilità delle istruzioni di entrata-uscita permette la creazione di validi programmi interattivi e una razionale realizzazione della stampa dei risultati.

Il linguaggio "C"

"C" è un linguaggio di programmazione estremamente completo, adatto a quasi tutti gli scopi. E' stato sviluppato sul sistema operativo Unix e sia Unix che il suo software sono scritti in "C".

di M.L. Nitti - D.R. Matturro

"C" può essere classificato come un linguaggio a basso livello; questo fatto non conferisce un aspetto negativo, ma significa che esso permette di interagire con la macchina come un linguaggio Assembler. Può maneggiare indirizzi e gestirli con gli operatori logici e aritmetici normalmente implementati sui microprocessori.

Un programma in "C" si presenta come una serie successiva di moduli, il primo dei quali si chiama sempre "main()" e gli altri hanno nomi scelti dall'utente. Prima di ciascuna routine possono esserci una serie di definizioni di costanti e una serie di dichiarazioni di moduli da "attaccare" al programma. Vediamo un esempio di programma completo.

La simbologia, come si può notare, è incomprensibile per i non iniziati. Il programma visto calcola la lunghezza di una stringa (che al massimo contiene 1000 caratteri) e la stampa. In questa sede non avrebbe senso spiegare ogni punto del programma, ma, a titolo di curiosità, e soprattutto per dimostrare quanto è preciso il "C", citiamo solamente gli operatori di incremento e decremento. Essi sono rappresentati da: ++ e --. Scrivere ++n oppure n++ equivale alla ben conosciuta frase n=n+1. Lo stesso vale per --.

```
# include <stdio.h>
# define MAXLINE 1000
main ()
{
     char line{MAXLINE};
     while(getline(line,MAXLINE) > 0)
          printf("%s", line);
}
getline(S, lim)
char s{};
int lim;
{
    int c,i;
    i = 0;
    while (--lim > 0 && (c=getchar ()) !=EOF && c != '\n')
    S[i ++] = c;

    if ( c = '\n')
       s{i++} = c;
    s{i} = '\0';
    return(i);
}}
```

La particolarità è che:

++n incrementa n prima di usarlo;
n++ incrementa n dopo averlo usato.

"C" è relativamente "piccolo" e può essere descritto in un tempo relativamente breve. I compilatori "C" sono semplici e compatti e tra di loro molto più compatibili delle cosidette versioni standard del Fortran.

Il solo tipo di dato è, fondamentalmente, la parola di macchina: si accede ad altre specie di "oggetti" mediante operatori oppure funzioni. I tipi di dati definibili sono i caratteri, gli interi di diverse ampiezze, i numeri reali. In aggiunta, c'è una gerarchia di dati derivati creati con puntatori: arrays, strutture, unioni, funzioni.

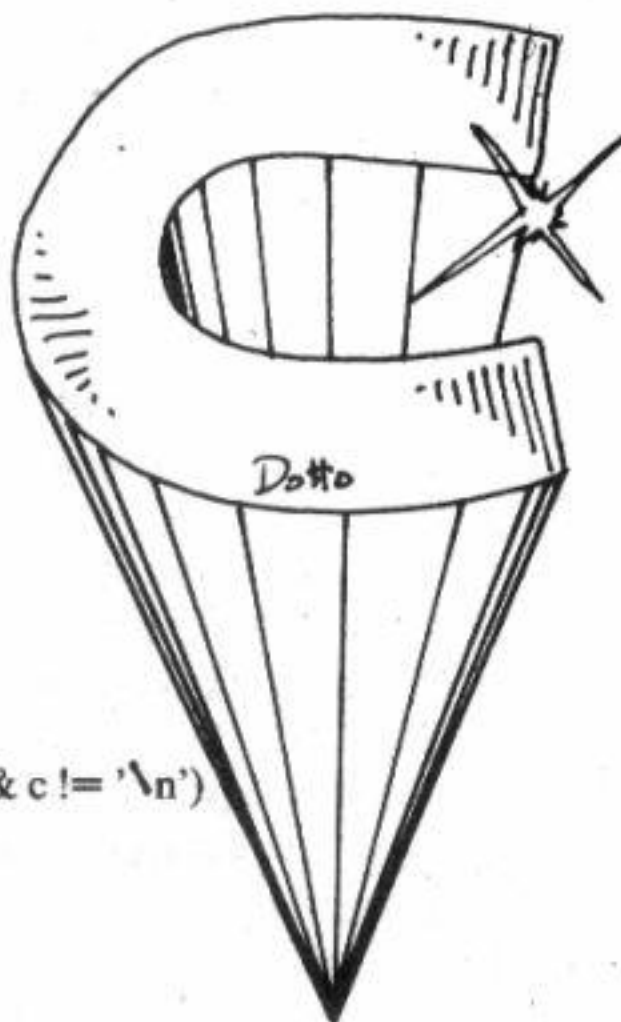
"C" fornisce le fondamentali istruzioni di controllo per la costruzione di programmi ben strutturati:

- per prendere decisioni (if);
- loop con test all'inizio (for, while);
- loop con test alla fine (do);
- scelta fra più casi possibili (switch).

Non fornisce invece alcuno strumento per Input/output. Non esistono frasi tipo READ, WRITE e metodi di accesso ai file. Tutte queste operazioni devono essere svolte da funzioni. Analogamente tutte le funzioni sull'analisi delle stringhe di caratteri devono essere create pazientemente dall'utente.

Il linguaggio "C" viene usato per gli scopi più svariati: creazione di sistemi operativi, calcolo numerico, creazione di word-processor e di data-base.

"C" è sicuramente un linguaggio che offre molto a chi molto gli si dedica: bisogna lavorare parecchio per sfruttarlo bene. E' eccezionale per lavori ad alto livello, consigliabile a persone con una precedente esperienza di linguaggi.

E' invece assolutamente sconsigliabile come primo linguaggio di programmazione oppure per i principianti.

Lisp & Logo

Da Lisp a Logo: ovvero l'intelligenza artificiale a cavallo di una tartaruga.

di M.L.Nitti - D.R. Matturro

Forse non tuttisanno che.... il "linguaggio della tartaruga" è uno dei più potenti e moderni tra quelli creati e usati negli ambienti dell'AI (Artificial Intelligence).

Per fare un po' di storia e giustificare quanto appena affermato, Logo nasce a Cambridge nel Massachussets ad opera di alcuni ricercatori della società Bolt Beranek e Newman e si sviluppa, presso l'università di quello stato, nei laboratori del Mit (Massachussets Institute of Technology).

In questo laboratorio si sono formati moltissimi degli scienziati che si occupano di AI, ovvero una branca sperimentale dell'informatica dove si sa possibilità di riprodurre, servendosi di un computer, i processi cerebrali.

Proprio quel Logo che molti sono portati a sottovalutare, perchè visto solo sotto l'aspetto grafico della tartaruga, è stato usato per sperimentare progetti di simulazione del pensiero umano. Chiaramente non ci riferiamo a programmi dotati di "personalità" o capaci di comportamenti completamente autonomi e altre diavolerie che si vedono nei film di fantascienza, ma a ricerche più terrene: dai programmi che giocano a scacchi a quelli in grado di formulare piccole teorie matematiche e altri temi del genere.

Ci troviamo quindi di fronte a un linguaggio completo capace di assolvere alle stesse funzioni del Basic, ma maggiormente dotato, rispetto a quest'ultimo, sia nella grafica, sia nella gestione dei dati, sia nelle istruzioni di controllo.

Logo e Lisp

Un linguaggio più potente e più facile del Basic. Allora perchè non dotarne tutti gli "home"? Sarebbe stato infatti molto meglio, ma purtroppo quando le scelte vanno fatte in funzione della memoria occupata, non è certo Logo a essere la soluzione migliore.

Quest'ultimo, infatti, occupa da solo circa 30 Kbyte di memoria e necessita ovviamente di ulteriore spazio per funzionare correttamente, contro gli 8K, tanto per citare un personal, dell'interprete Basic del Commodore.

I più curiosi saranno lieti di sapere che il C-64 può avvalersi di un interprete Logo caricabile da disco. Questo interprete è la traduzione in italiano del Logo originale, quello nato al Mit di chiara derivazione del Lisp.

E' proprio il Lisp la matrice del Logo e infatti le caratteristiche che fanno di quest'ultimo un linguaggio spiccatamente evoluto sono le stesse che fanno del Lisp il linguaggio della quarta generazione.

Liste di dati

Logo gestisce i dati sotto forma di liste, proprio come il Lisp. Questo non significa che non esistono le variabili, ma semplicemente che gli "insiemi" organizzati

di variabili (quali vettori e matrici) si ripresentano qui in una nuova formula meglio utilizzabile per costruire basi di dati, ovvero strutture di informazioni.

I dati strutturati in liste e trattati come tali sono meglio reperibili in quanto è più facile estrarne solo la parte che ci serve, con estrema velocità.

Vediamo che cosa significa.

Una base di dati potrebbe essere uno schedario contenente informazioni di vario tipo su tutti gli allievi di una scuola.

Ciò corrisponde anche a quello che comunemente chiamiamo archivio, ovvero un insieme di file-dati relativi a uno o più argomenti organizzati tra loro. Se tale archivio fosse ben organizzato, noi potremmo reperire facilmente le informazioni che ci interessano utilizzando un linguaggio di interrogazione, ovvero una serie di comandi che permettano l'accesso a un gruppo di dati secondo determinate condizioni.

Ad esempio, volendo conoscere il nome degi allievi maggiorenni, che frequentano il secondo anno, in possesso di un computer, se la base di dati fosse ben architettata e contenesse queste informazioni, gli allievi del secondo anno rapresenterebbero il gruppo di dati cui si intende giungere e la maggiore età (unita al possesso di computer) la condizione.

Ricorsione, questa sconosciuta

Viste così le cose, sembrerebbe di aver fatto la scoperta dell'acqua calda: qualsiasi Data Base che si rispetti espleta queste funzioni. Solo che un Data Base si ferma a questo punto mentre, al contrario, per Lisp e Logo questa rappresenta la "base" da cui partire.

I due linguaggi possiedono (il Lisp in forma più potenziata) una serie di istruzioni di controllo che permettono di manipolare le liste secondo tecniche avanzatissime.

Senza voler insegnare in questa sede l'uso di queste "alchimie", ne facciamo un breve cenno. In primo luogo ogni procedura può richiamare se stessa, come se, all'interno di una subroutine del Basic, inserissimo un GOSUB alla stessa linea di programma. Il Basic, in tal caso, ci darebbe un errore di OUT OF MEMORY, perchè tale possibilità non è prevista.

Logo esegue tranquillamente il comando entrando in un sottolivello e, in caso di ulteriore chiamata, in un sotto-sottolivello e così via scendendo e risalendo a piacere.

Questo gioco di scatole cinesi si chiama ricorsione ed è molto difficile da "digerire" teoricamente, anche perchè si sa dove inizia ma non dove finisce. Con l'uso, però, i concetti esposti risultano estremamente chiari.

L'altra prerogativa è rappresentata dalle variabili locali. Ad esempio la variabile "A" di una procedura mantiene il valore assunto a ogni livello e, di conseguenza, se al livello 2 tale variabile valeva 4, quando rientriamo in quel livello la ritroveremo ancora con questo valore.

Per chiarirci le idee, è un po' come se avessimo un vettore A(n) dichiarato, però senza indice e quindi senza tutte le complicazioni di incremento e decremento che gli indici comportano.

Ma chiudiamo la "scatola" delle scatole cinesi e prepariamoci a una nuova sopresa.

Un linguaggio personalizzato

La cosa più simpatica di Logo è l'autoapprendimento.

Non fraintendiamo: non è che Logo impara da solo nuove cose, ma, semplicemente, è possibile aggiungere nuovi comandi a quelli già esistenti. Si può realizzare, in tal modo, una vera e propria biblioteca personale utilizzabile nello stesso modo in cui si usano le istruzioni.

Schiariamoci le idee con qualche esempio:

- PS è un'istruzione che esegue la pulizia dello schermo;
- STAMPA corrisponde al print del Basic con diversi parametri;
- SOMMA-DI esegue la somma dei valori o variabili che seguono.

I comandi seguenti ottengono l'effetto di pulire lo schermo, stampare la frase "ecco la somma richiesta", eseguire la somma delle variabili "A" e "B" e stamparne il risultato:

```
PS
STAMPA [ECCO LA SOMMA DI
A+B]
STAMPA SOMMA-DI A B
```

Tale gruppo di istruzioni poteva diventare una procedura se avessimo scritto i comandi stessi dopo aver richiesto all'interprete di considerarli come tale. Ovvero sarebbe stato sufficiente scrivere "PER" seguito dal nome da assegnare alla procedura, per esempio "STAMPASOMMA":

```
PER STAMPASOMMA
PS
STAMPA [ECCO LA SOMMA DI
A+B]
STAMPA SOMMA-DI A B
FINE
```

Dopo tale operazione, semplicemente digitando:

```
STAMPASOMMA
```

avremo ottenuto lo stesso effetto della sequenza di istruzioni da cui eravamo partiti.

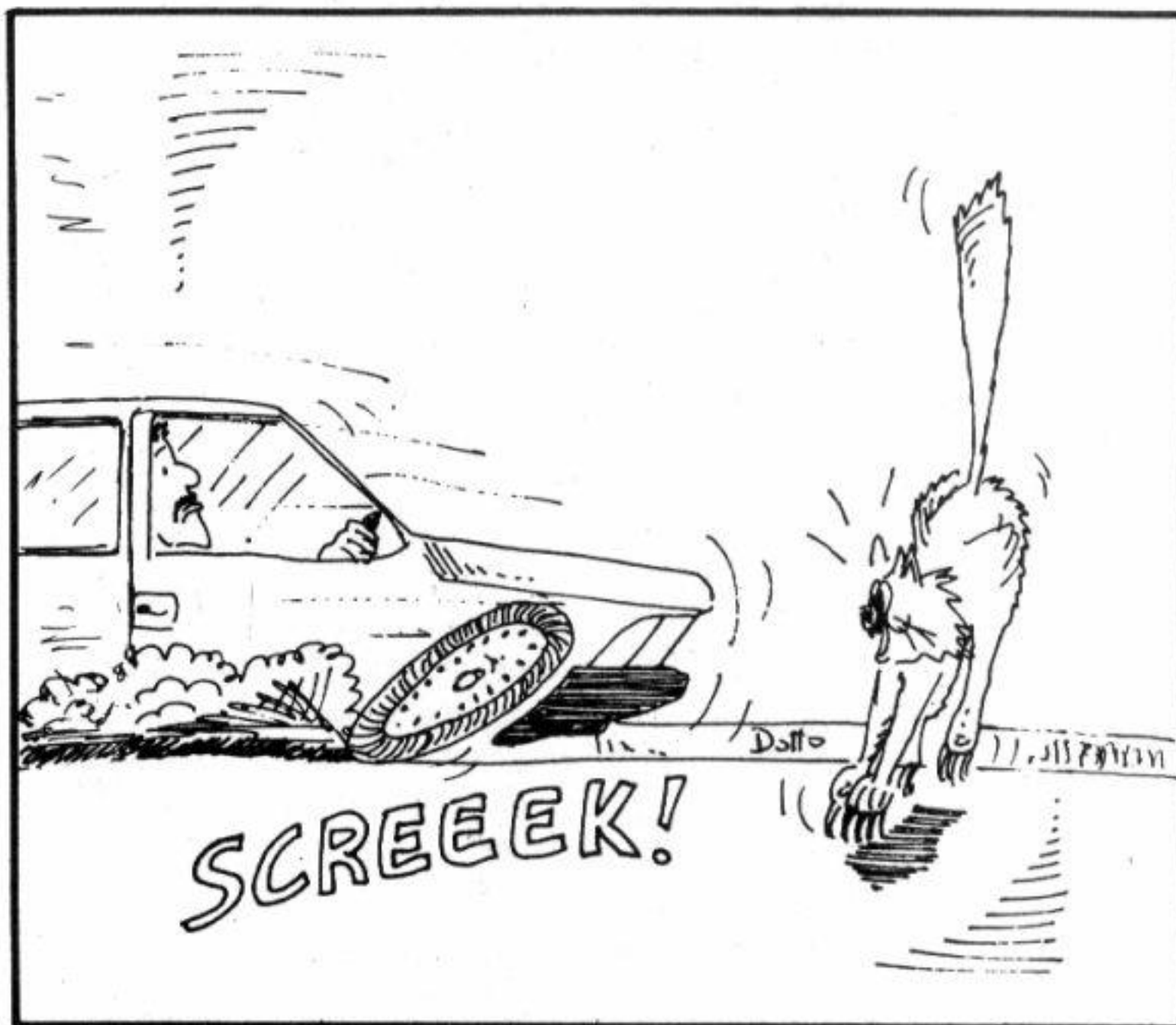
Una volta definita come procedura, STAMPASOMMA diventa un'ulteriore parola del vocabolario di Logo utilizzabile come istruzione in qualsiasi altra procedura che potrebbe diventare istruzione a sua volta.

Qui si rischia di tornare alle scatole cinesi e qualcuno potrebbe pensare che il Logo sia stato inventato appunto dai cinesi per fare impazzire gli occidentali. Invece Logo è una "macchina" di facilissimo uso e di grande flessibilità, tanto che ogni utente ha la possibilita di personalizzare il sistema facendogli appunto "apprendere" nuovi comandi congeniali al proprio lavoro di programmazione.

Alla fine di una sessione di lavoro, con un semplice comando, possiamo salvare la nostra biblioteca di comandi e ricaricarla in seguito arricchendola, magari, finchè memoria permette.

Venti domande sul linguaggio macchina

Rispondiamo in queste pagine alle numerose domande, tutte sul linguaggio macchina (LM), giunte in redazione. Un modo semplice ed efficace per accostarsi senza tanti problemi a quest'argomento.

di Alessandro de Simone

☐ **Il LM è un vero e proprio linguaggio? Esiste un LM standard così come esiste un Basic standard?**

• All'interno di un qualsiasi computer è presente un circuito elettronico, il microprocessore (detto spesso semplicemente "micro") che, per funzionare, richiede una successione di codici rappresentanti il linguaggio del micro stesso. Questi codici, (che sono semplici numeri interi il cui valore è compreso tra 0 e 255) sono molto difficili da comprendere, soprattutto da chi conosce appena il Basic. Inoltre, se non bastasse, non è sufficiente conoscere il linguaggio di un certo microprocessore per utilizzare programmi in LM di tipo "universale".

Per esempio, se, con *qualsiasi* computer che parli il Basic, digitate un comando banale come:

PRINT"PIPPO"

la conseguenza sarà la visualizzazione della parola "PIPPO" sul video del computer. Ciò, ripetiamo, sia che il computer sia un Vic-20, un C-64 oppure un Ibm eccetera. Il motivo dell'identico comportamento è dovuto al fatto che il Basic è, almeno in generale, un linguaggio universale, cioè standard. E' ovvio, quindi, che il computer viene, dai fabbricanti, "adattato" al linguaggio standard, dato che non deve avere alcuna importanza il nome della marca del calcolatore, ma solo il linguaggio adoperato.

Nel caso del linguaggio del microprocessore (LM, appunto) le cose stanno ben diversamente. Infatti, il linguaggio dei micro installati nel Vic-20, nel C-64, nel C-16, nel Plus/4 e nei grossi PET, è sempre lo stesso, dato che sempre la stessa è la "famiglia" cui appartengono i micro impiegati: la 65XX. Ciò significa che, a parte alcuni casi particolari, il micro di base è sempre il 6502 (grossi PET, Vic-20), che diventa 6510 nel C-64 e 7510 nel

C-16 e Plus/4. In altre parole, il codice 169 dell'istruzione "LDA" (su cui torneremo tra breve) significa la stessa cosa su ognuno dei computer menzionati. Se proviamo, però, a utilizzare un programma scritto in LM espressamente per il Vic-20 e cerchiamo di farlo girare su macchine diverse (es. C-64), abbiamo la sgradita sorpresa di non vederlo funzionare, nonostante il linguaggio sia esattamente lo stesso. Il più delle volte, anzi, sarà necessario spegnere il computer e riaccenderlo a causa del pasticcio combinato.

Vediamo di spiegarci un'apparente contraddizione:

- programmi scritti in linguaggi evoluti (come il Basic) sono eseguibili su qualsiasi computer (pur se dotati di micro completamente diversi tra loro);
- programmi in LM (linguaggio realmente standard perchè è il linguaggio specifico di un micro ben definito) funzionano solo su un particolare modello di calcolatore e su nessun altro, anche se dotato dello stesso identico microprocessore.

Per venirne a capo, riferiamoci a un esempio banale. Sapete benissimo che, in una qualsiasi automobile, vi sono tre pedali, posizionati da sinistra a destra: frizione, freno, acceleratore.

Quando l'automobilista preme il freno, ottiene lo stesso identico effetto (decelerazione e arresto della vettura) qualunque sia l'automobile manovrata.

Ciò significa che se i freni sono a disco, a tamburo, idraulici o meccanici, serviti oppure no, l'effetto è sempre lo stesso.

Profonde differenze esistono, però, se consideriamo "intimamente" la reale struttura dell'impianto frenante dei numerosi modelli in circolazione.

La disposizione dei pedali può esser paragonata ai comandi Basic che, indipendentemente dal micro installato, producono lo stesso effetto finale.

I tubi dell'olio, le dimensioni dei dischi e delle pastiglie e altri particolari meccanici possono (a grandi linee) esser paragonati al LM che, funzionante in una determinata auto, non funzionano se trasferite senza modifiche su un'altra auto per intuibili motivi: diverso peso della vettura, diversa lunghezza dei tubicini dell'olio, diverso diametro dei dischi, diverso posizionamento dei leveraggi dei pedali, eccetera.

Insomma il Basic è quella parte del calcolatore che, aiutata dal sistema operativo (SO) utilizza razionalmente le capacità del microprocessore.

In conclusione: benchè il linguaggio macchina sia più universale del Basic (considerando le istruzioni del micro) un programma scritto in LM per un modello differente, risulta di difficile adattamento al proprio calcolatore.

□ Che differenza c'è tra Basic e LM?

- Il Basic, come qualsiasi altro linguaggio "evoluto", rappresenta un "filtro" tra l'utilizzatore del calcolatore e il micro installato nel computer.

Ad esempio, sapete benissimo che cosa accade se scattate una foto utilizzando una moderna macchina fotografica del tipo a sviluppo immediato. E' sufficiente inquadrare e scattare: subito dopo fuoriesce la foto già fatta.

Non è certo la pressione del vostro dito a compiere il miracolo! Lo scatto, infatti, non fa altro che mettere in moto una serie di complicate procedure tecnico/chimiche legate insieme da delicatissimi equilibri. Tutto il lavoro di sviluppo e stampa è stato compiuto, in precedenza, dai tecnici della casa fotografica progettando e producendo, con grande accuratezza i prodotti da voi acquistati.

La macchina fotografica e la pellicola, in definitiva, rappresentano tutto il lavoro compiuto "a monte" proprio per fare in modo che voi, con la semplice pressione di un dito (e pagando...) possiate ottenere l'immagine desiderata.

Più è semplice l'uso di un apparecchio sofisticato, più contiene tecnologia.

Gli oggetti ad alto contenuto tecnologico (computer, televisori, automobili) sono, appunto, complessi pur se semplici da usare. Al contrario, quelli a basso contenuto di tecnologia (matite, sedie, lampadine) sono semplici da realizzare.

Un linguaggio Basic da 8K vale meno di uno da 16K (a patto che siano ben strutturati entrambi). Analogamente un vecchio calcolatore, di quelli che funzionavano solo con la programmazione in LM, è a basso contenuto... di linguaggio rispetto a uno che consente di lavorare anche con linguaggi evoluti.

Un programma scritto in LM è scritto secondo la sintassi (un po' complicata) del microprocessore. Un programma in Basic, invece, sfrutta tutte le comodità (segnalazione di errori, semplicità di espressioni, calcoli complessi risolti in modo semplice) introdotte dal fabbricante per render semplice l'uso del computer.

Vediamo un ultimo esempio per meglio comprendere la differenza tra Basic e LM.

E' possibile avviare il motore di un'auto con un semplice gesto della chiave di accensione.

Se, però, la batteria è scarica, la chiave non serve più ed è necessario che la vettura sia avviata a spinta. Ebbene il Basic rappresenta, in un computer, ciò che la chiave rappresenta per l'auto: la comodità di utilizzo. Ciò non toglie che si possono ottenere effetti analoghi ricorrendo ad altri sistemi che, in un modo o nell'altro, riescano egualmente a far funzionare il cuore del sistema (motore oppure micro a seconda dei casi).

□ Il LM ha una struttura più semplice o più complessa del Basic?

- Il modo di programmare è decisamente più complicato a causa del numero apparentemente elevato di istruzioni LM necessarie per realizzare "qualcosa". La struttura logica, comunque, è molto simile a quella del Basic.

Vi sono istruzioni di salto (paragonabili al GOTO), di subroutine (GOSUB) di scelte condizionate (IF...) di operazio-

ni algebriche (+ -), operazioni logiche (AND, OR) e così via.

Mentre, però, il Basic offre la possibilità di utilizzare righe di programma numerate ed è possibile inserire più istruzioni e comandi in una sola riga, col LM bisogna scrivere una sola istruzione "per rigo" (che rigo, vero e proprio, non è), trascrivere manualmente (munendosi, soprattutto agli inizi, di carta e penna) indirizzi e altre cose "importanti".

Chi conosce in modo decente il Basic può certamente accostarsi al LM, dato che la logica di programmazione è straordinariamente simile.

□ Quali vantaggi si ottengono da programmi scritti in LM piuttosto che in Basic?

□ Quali inconvenienti comporta la programmazione in LM?

• Abbiamo detto che il Basic si "interpone" tra il microprocessore e l'utente. Il vantaggio è una maggior semplicità d'uso, lo svantaggio è rappresentato da una certa lentezza di esecuzione.

E' come se, per mangiare un piatto di spaghetti, una bistecca e un frutto, decidessimo di andare al ristorante o di prepararci un pranzo a casa nostra. Nel primo caso si hanno innegabili vantaggi (comodità di non cucinare, mancanza di piatti da lavare, eccetera). Nel secondo caso, a parte il risparmio di denaro, si ha un evidente risparmio di tempo dovuto al fatto che non è necessario spostarsi da casa e che è realmente possibile cucinare a tempo di record le semplici vivande indicate. Nel primo caso occorre almeno un'ora; nel secondo, invece, in meno di mezz'ora possiamo soddisfare la nostra fame.

Chi ha provato a realizzare in Basic giochi in cui si verificano animazioni (movimenti di sprite o di caratteri) mentre una musica, magari, suona in sottofondo, ha sicuramente notato i limiti imposti dalla lentezza del Basic. Questo linguaggio, infatti, prima di eseguire un qualunque comando, esegue *sempre* (anche se, in realtà, non sarebbe necessario) *tutti* i controlli tipici del linguaggio: esame della corretta sintassi, esame della posizione del cursore, esame delle locazioni video, eccetera.

Se fosse possibile aggirare gli ostacoli dei controlli (che risultino realmente superflui), il programma girerebbe molto più in fretta. Ebbene, programmando in LM si raggiunge proprio quest'obbiettivo: far eseguire al microprocessore soltanto le istruzioni necessarie evitando accuratamente quelle realmente superflue.

E' ovvio che è compito del programmatore individuare accuratamente i controlli superflui e stabilire con molta attenzione gli "ostacoli" aggirabili.

Il pericolo, infatti, c'è ed è grandissimo: in Basic, se sbagliamo a digitare un'istruzione, viene emesso un tipico messaggio (SYNTAX ERROR o altri); lavorando invece in LM il microprocessore, programmato male, prende lucciole per lanterne e... può accadere di tutto, anche il blocco totale della macchina che è necessario spegnere. In questi casi, oltre al danno determinato dal tempo dedicato alla stesura del programma, rimane l'amarezza di non sapere il perchè del blocco.

Il computer comunque non subisce danni di nessun tipo, qualunque sia l'errore commesso.

□ Con quali computer è possibile scrivere programmi in LM oltre che col Basic?

□ E' possibile scrivere un programma LM servendosi del Basic?

□ A che servono le istruzioni POKE, PEEK, SYS?

□ Perchè risulta complicato (per non dire impossibile) effettuare modifiche a programmi in LM scritti per altri computer, in modo da farli funzionare su calcolatori differenti?

• Un programma in LM, come già detto, è rappresentato da una successione di numeri che hanno, ciascuno, una simbologia ben precisa per il microprocessore.

Per esemplificare il concetto, facciamo un semplice paragone tra un comando Basic e uno simile (ma non proprio eguale) al comando LM.

A$="PIPPO" rappresenta, come è noto, la definizione di una stringa. Alla variabile A$ viene associata, col comando precedente, la stringa alfabetica formata dai caratteri "PIPPO". Possiamo anche dire che, con A$="PIPPO", si memorizza nella RAM del calcolatore la parola "PIPPO".

In LM i seguenti numeri (169 80 141 0 192), scritti in successione, rappresentano, per il micro 6502, un ordine analogo: quello di memorizzare il codice ASCII della lettera "P", cioè 80, in una ben definita cella di memoria.

Vediamo in dettaglio che cosa rappresentano esattamente i cinque numeri-codice prima indicati.

169 80 Considera il numero 80...
141 0 192 ...e trascrivilo nella locazione di memoria n.192*256+0=49152

Avrete intuito che il numero 169 rappresenta l'ordine:
"prendi" il numero che viene subito dopo (in questo caso 80).

Allo stesso modo il numero 141 significa, per il micro:
scrivi il numero che hai appena "preso", nella locazione di memoria il cui indirizzo va ricavato dai due numeri che vengono subito dopo il numero 141.

Che cosa significa, ad esempio, la seguente successione di numeri:
169 65 141 50 192

E' semplice: prendi (169) il numero 65 e scrivilo (141) nella locazione il cui indirizzo è dato dal calcolo: 50+192*256, cioè 49202.

Torniamo alla domanda iniziale: con quale computer si può scrivere un programma in LM? Con qualsiasi computer, purchè consenta di scrivere nella memoria RAM, si è in grado di utilizzare programmi in LM.

Tutti i calcolatori Commodore hanno le istruzioni per compiere il miracolo:

POKE, che consente di "scrivere" un qualsiasi codice macchina in qualsiasi locazione di memoria;

PEEK che consente di leggere il contenuto di una qualsiasi cella di memoria;

SYS che consente di attivare un qualsiasi programma in LM.

Vediamo alcuni esempi.

Esaminiamo il seguente programma Basic, funzionante solo sul C-16 e commentiamolo insieme. Subito dopo vedremo le altre versioni per il Vic-20 e per il C-64.

```
90 REM COMMODORE C-16
100 POKE 819,169
110 POKE 820,1
120 POKE 821,141
130 POKE 822,0
140 POKE 823,12
150 POKE 824,96
160 PRINT CHR$(147)
170 PRINT
180 SYS 819
```

100 - Nella locazione 819 viene trascritto, grazie all'istruzione POKE, il numero 169 che, come ricorderete, è il primo codice del mini programma scritto in LM.

Perchè, fra le tante, è stata scelta proprio la locazione 819?

Dovrebbe esser noto che nel computer vi sono memorie di tipo ROM e RAM. Nelle prime non è possibile "scrivere" e, di conseguenza, è necessario escluderle per i nostri scopi. Alcune locazioni RAM, inoltre, sono riservate al sistema operativo e, benchè sia possibile scrivervi mediante istruzioni POKE, è bene evitare di servirsene, altrimenti si rischia di "inchiodare" il sistema. Dobbiamo tener conto, ancora, che se scriviamo un programma Basic, questo occupa; man mano che viene digitato (oppure caricato da nastro o da disco) una zona di memoria RAM ben precisa, che nel C-16, ad esempio, parte dalla locazione 4096 e occupa, in totale, una zona di memoria proporzionale alla lunghezza del programma stesso. Tentare di scrivere qualcosa nella zona del Basic significa cancellarlo in parte e, in ogni caso, alterarne il contenuto.

Per brevi programmi, allora, rimane una zona di memoria RAM che serve abitualmente al computer per il "colloquio" col registratore. Questa zona (a disposizione dell'utente a patto che, durante l'uso, non venga usato il registratore) è un blocco di RAM che, nel C-16, è numerato da 819 a 1010.

Nel C-64 e Vic-20, invece, la zona RAM dedicata al registratore (detta, più propriamente, "cassette buffer") è numerata da 828 a 1019.

110-150 - Vengono trascritti uno per uno, nella zona RAM indicata, i codici macchina che rappresentano, ciascuno, un comando (169, 141) oppure un "argomento" del comando stesso (0, 12). L'ultimo codice (96) significa, in LM, un "ritorno" al Basic.

160-170 - Cancella lo schermo e posiziona il cursore un rigo più in basso.

180 - SYS è il comando Basic che impone al computer di "entrare" in linguaggio macchina "abbandonando", ovviamente, lo stesso linguaggio Basic. Da questo momento il microprocessore, anzichè eseguire il programma Basic, eseguirà il programma in linguaggio macchina, che inizia a partire dalla locazione numerata col valore della SYS che, nel nostro caso, è proprio 819. Ma in 819 è presente prorio la prima istruzione LM trascritta con la POKE di riga 100 e il microprocessore eseguirà il programma. Vediamo quale risultato si ottiene.

169 1 "prende" il numero 1.

141 0 12 "scrive" il numero uno considerato in precedenza (169 1) e lo "depositа" (141) nella locazione 12*256+0, cioè nella cella 3072. Ma questa è una cella della memoria del video del C-16 e il risultato è simile a quello ottenibile in Basic con POKE3072,0. Compare, cioè, il carattere "A" in alto a sinistra sul video.

96 subito dopo aver depositato il numero 0 nella locazione 3072, questo codice impone al microprocessore di "restituire" il comando al Basic.

Il listato, però, è complicato e può esser sicuramente semplificato ricorrendo alle istruzioni READ... DATA. Ecco, in questo caso, un modo per trascrivere un programma LM con i tre computer della Commodore:

```
100 REM C-16
110 FOR I=0 TO 5
120 READ A
130 POKE 819+I,A:NEXT
140 DATA 169,1,141,0,12,96
150 PRINT CHR$(147)
160 PRINT
170 SYS 819
```

```
100 REM C-64
110 FOR I=0 TO 5
120 READ A
130 POKE 828+I,A:NEXT
140 DATA 169,1,141,0,4,96
150 PRINT CHR$(147)
160 PRINT
170 SYS 828
```

```
100 REM VIC-20 INESPANSO
110 FOR I=0 TO 5
120 READ A
130 POKE 828+I,A:NEXT
140 DATA 169,1,141,0,30,96
150 PRINT CHR$(147)
160 PRINT
170 SYS 828
```

Nel Vic 20 (come pure in alcuni antiquati esempari di C-64) è necessario colorare la cella video prima di eseguire il programma. In questi casi è necessario premere il tasto Home (senza Shift), dopo aver fatto girare il programma, per veder lampeggiare la "A" sotto il cursore.

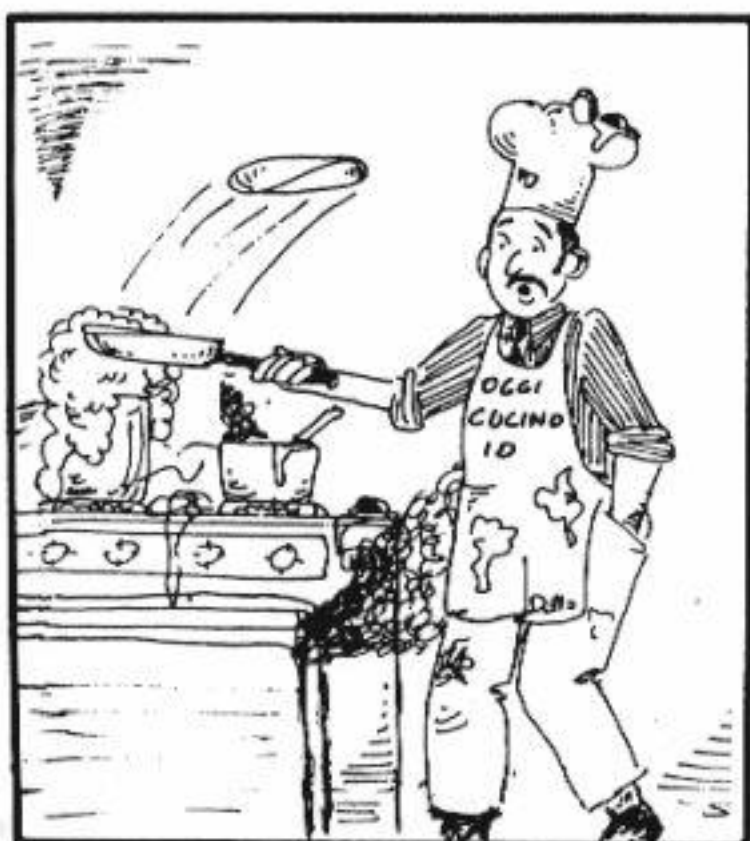
Avrete notato sicuramente che, per ottenere lo *stesso* effetto sui tre computer (visualizzazione del carattere "A" nella prima cella in alto a sinistra dello schermo), è necessario "personalizzare" il programma in LM benchè la sintassi sia sempre la stessa. Fondamentale, inoltre, è la stessa struttura della memoria del calcolatore.

Se, per fare apparire semplicemente un carattere sullo schermo, è necessario effettuare modifiche al listato LM, figurarsi il lavoro da svolgere per adattare programmi interi lunghi migliaia di codici-macchina!

E pensare che si ottiene lo stesso effetto, sui tre computer, col semplice listato Basic:

100 PRINT CHR$(147)"A"

E' ovvio che, per fare apparire un carattere sullo schermo non è consigliabile servirsi del LM.

Vedremo, nei prossimi numeri di Commodore Computer Club, che in altri casi il LM è realmente più utile del Basic.

☐ **Come si può scrivere un programma in LM?**

☐ **Che cosa è l'Assembler?**

☐ **Che cosa sono i codici mnemonici del LM?**

• Abbiamo visto che un programma LM (che altro non è se non una successione di numeri interi compresi tra zero e 255), può esser semplicemente trascritto in un'area RAM mendiante un programma Basic che, sfruttando adeguatamente le istruzioni READ, DATA, POKE, FOR, NEXT, legge i vari codici e li "poka" nelle locazioni desiderate.

Un altro sistema consiste nell'utilizzare programmi specifici che aiutano a digitare i codici LM.

Nel C-16 (e nel Plus/4), ad esempio è incorporato il programma Monitor. Per il C-64 e il Vic 20 sono disponibili programmi analoghi che, però, è necessario caricare da nastro o disco per utilizzarli. Nei prossimi numeri di Commodore Computer Club diremo, un po' per volta, come utilizzare i vari programmi Monitor.

Un altro sistema, molto più valido, è quello di utilizzare specifici programmi Assembler che aiutano moltissimo a stendere un programma in LM.

Precisiamo anzitutto che per programma Assembler si intende un programma scritto mediante un programma compilatore chiamato Assembler. Quest'ultimo è un'utility che facilita la stesura di un programma LM perchè ricorre a istruzioni simboliche molto più facili da ricordare che non i codici numerici del linguaggio macchina.

Consideriamo, ad esempio, il programma "puro" in LM di prima:

169 1 141 0 12 96

Anzichè scriverlo come un gruppo ininterrotto di cifre, sarebbe meglio scriverlo nel modo seguente, raggruppando tra loro, cioè, i codici che si riferiscono ad una singola istruzione:

```
169 1      Considera il numero 1
141 0 12   "Scrivilo" nella locazione
12*256+0
96         Ritorna al Basic
```

Lo stesso programma, scritto in Assembler, diventa enormemente più chiaro:

```
LDA 1
STA   3072
RTS
```

LDA è il codice mnemonico (derivante dall'inglese Load Accumulator=carica nell'accumulatore) che indica l'ordine di caricare (scrivere) nell'accumulatore (memoria speciale presente all'interno del micro) il numero che segue (nel nostro caso particolare, 1).

STA è il codice mnemonico (STore Accumulator) che impone di memorizzare il contenuto dell'accumulatore nella locazione il cui indirizzo è indicato dal numero che segue (proprio 3072) senza che sia necessario effettuare scomode e noiose "traduzioni" (12*256+0=3072).

RTS significa ReTurn from Subroutine (ritorna dalla subroutine). Nel caso specifico significa: ritorna al Basic.

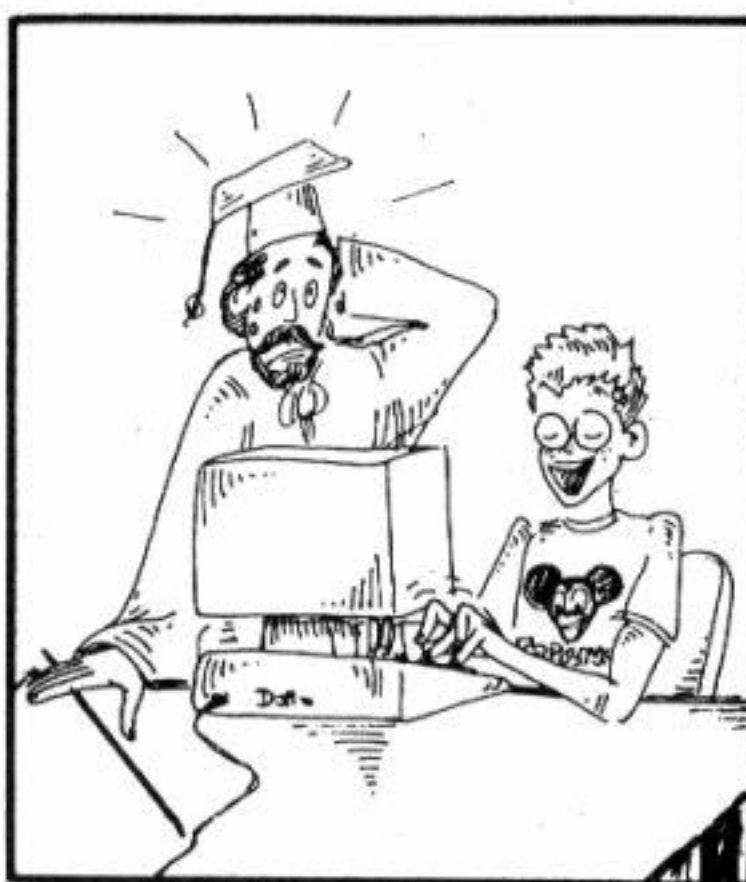
Naturalmente i codici mnemonici sono molti di più e alcuni di essi possono assumere significati differenti a seconda dei caratteri posti subito dopo di loro (virgola, punto, dollaro, eccetera).

Il linguaggio Assembler, che è un compilatore, è un compendio validissimo per chi vuole programmare in LM perchè, al vantaggio di ottenere un programma scritto in LM, aggiunge quello di renderlo comprensibile e "scorrevole" anche ai meno esperti.

☐ **E' necessario esser molto bravi in matematica per imparare il LM oppure l'Assembler?**

• Nel campo dell'informatica è indispensabila avere pazienza e frenare l'im-

pulso di bruciare le tappe per imparare in fretta. Ragazzi giovanissimi, spesso più preparati ed esperti di personale laureato, hanno acquisito la propria esperienza non grazie a un'intelligenza "superiore", ma solo applicandosi con costanza al calcolatore. Molti, strano a credersi, odiavano la matematica prima di accostarsi al computer. Ci sono giovani, infine, che hanno dimostrato notevoli miglioramenti a scuola da quando hanno iniziato a smanettare sulla tastiera.

Nel campo del linguaggio macchina, credetemi, è sufficiente sapere le sole quattro operazioni. Al resto provvedono la fantasia e la serietà di chi lo utilizza...

□ Per utilizzare programmi in LM è necessario conoscerne il linguaggio?

• Niente affatto: sicuramente avrete caricato sul vostro computer, almeno una volta, un videogioco. Ebbene, sappiate che il 99% dei videogame sono scritti in LM, ma questo fatto non vi ha impedito di utilizzarli e di divertirvi.

Se vi accingete, al contrario, a digitarli, leggendoli da una rivista (come Commodore Computer Club) dovete prestare la massima attenzione a digitare i valori così come sono, senza sbagliare la trascrizione di un solo dato, che potrebbe esser fatale per il funzionamento del programma.

In genere, però, lo stesso programma contiene al suo interno un minimo di controllo che avverte l'utente che è stato commesso un errore di trascrizione.

□ E' possibile danneggiare il computer scrivendo un programma LM in modo errato?

• No, in nessun caso. L'errore più "grave" che si può commettere è il tentativo di scrivere un dato in una zona ROM invece che RAM. Anche se questa operazione viene eseguita milioni di volte (e può capitare se si incorre in un loop chiuso) nessun danno può esser provocato

alla ROM nè a qualsiasi altro circuito.

Se un programma LM è scritto male, oppure è mal progettato, può capitare che i numerosi tentativi effettuati per "riprendere" il controllo del computer (Run/stop e Restore, tasto di Reset eccetera) non sortiscano alcun effetto. In casi come questi l'unico modo per riprendere il lavoro consiste nello spegnere e riaccendere l'apparecchio (e le periferiche collegate) perdendo, però, il programma digitato. Ecco perchè è buona norma registrare *sempre* un programma LM oppure Assembler *prima* di mandarlo in funzione. Se non dovesse funzionare, non sarete costretti a riscriverlo ex-novo, ma sarà sufficiente ricaricarlo (da nastro o disco) ed esaminarlo attentamente in modo da rintracciare (ed eliminare) l'errore.

□ Perchè, per lavorare in LM, non è sufficiente la matematica che conosciamo?

□ Che cosa è la notazione binaria?

□ Che cosa è la notazione esadecimale?

□ Come si può convertire un numero decimale nel relativo binario ed esadecimale?

• Abbiamo detto prima che i numeri-codice di un programma in LM sono compresi tra zero e 255. Ciò è giustificato dal fatto che un microprocessore a otto bit (come appunto quelli montati sui computer Commodore), possono "trattare" otto simboli per volta, ciascunp dei quali può valere solo "1" oppure "0".

Vediamo di chiarire questo importante concetto, che rappresenta il fondamento dell'informatica.

Noi siamo abituati già alla notazione binaria (=due simboli soltanto) anche senza che ce ne rendiamo conto. Consideriamo, infatti, gli indicatori lampeggianti di direzione di una qualsiasi autovettura ed esaminiamo i vari casi che possono presentarsi:

• nessuna delle due lampadine lampeggia. Ciò significa che l'auto procede diritta (oppure è parcheggiata) e che l'automobilista non ha intenzione di svoltare;
• lampeggia il solo indicatore destro. Ciò significa che tra breve l'automobile girerà a destra;
• lampeggia l'indicatore sinistro. L'auto svolterà a sinistra;
• lampeggiano entrambe le lampadine. L'automobile è ferma in corsia di emergenza.

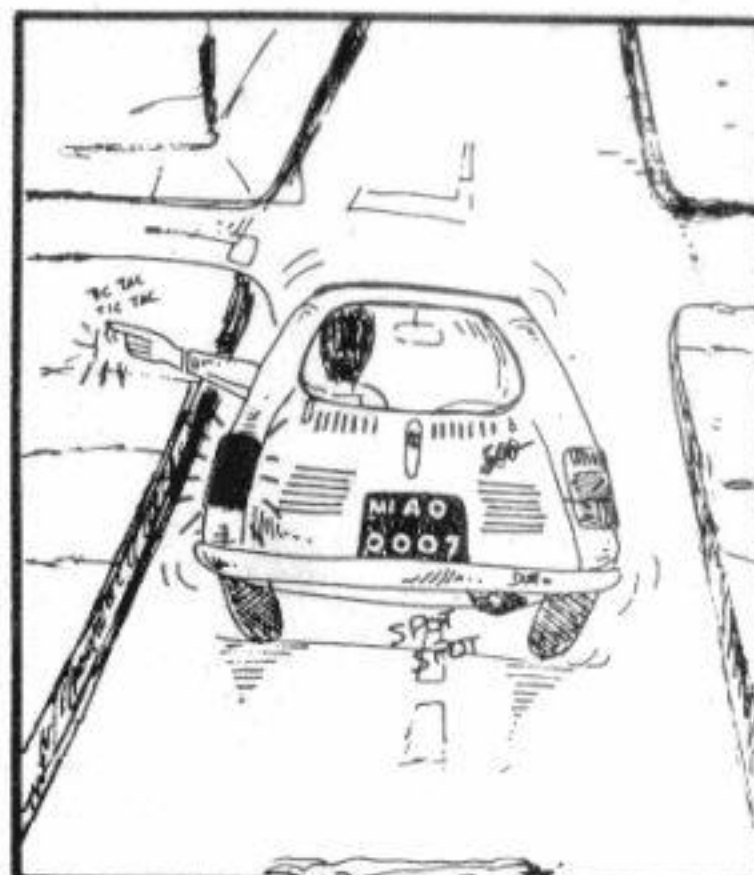
Con sole due lampadine (destra e sinistra) è possibile generare quattro "simboli" diversi.

Il numero quattro deriva dal numero due (le possibilità, cioè lo stato di "acceso" oppure "spento") elevato al quadrato (il numero delle lampadine).

Schematizzando l'esempio, le quattro possibilità possono essere indicate nel modo seguente:

Lampadine	
Sinis.	**Des.**
0	0
0	1
1	0
1	1

Nel caso di un semaforo, che è formato da tre lampade, sarebbe possibile generare un numero di due (stati possibili) elevato alla terza (numero di lampade), cioè otto simboli diversi. Schematizzando come prima si ha:

Lampade			Significato
Verde	Giallo	Rosso	
0	0	0	(Semaforo non funz.)
0	0	1	(Alt)
0	1	0	(Attenzione)
0	1	1	
1	0	0	(Via libera)
1	0	1	
1	1	0	(Affrettarsi)
1	1	1	

Come si può notare, è possibile individuare nel traffico stradale un numero inferiore di stati possibili.

Se invece di lampadine parliamo di bit e, invece di acceso e di spento, di "uno" e di "zero", il discorso di base sul linguaggio macchina non costituisce più un problema.

Riepilogando, avendo a disposizione otto bit (lampadine), ognuno dei quali può essere in uno solo di due stati (1, 0, oppure On, Off, eccetera) quante combinazioni è possibile ottenere?

2 elevato all'ottava potenza = 256 simboli diversi, proprio come abbiamo detto prima.

Il "primo" è costituito da un numero di otto zero (00000000), l'ultimo da un insieme di otto "uno" (11111111).

Il calcolatore, dunque, "ragiona" solo con gli zero e gli uno. Sarebbe un guaio se anche noi dovessimo scrivere un programma ricorrendo a due soli simboli. Fortunatamente c'è un sistema che consente di trasformare un qualsiasi numero binario (formato normalmente da otto bit) nel corrispondente decimale.

Vediamo un esempio.

A quale numero decimale corrisponde il numero binario di otto bit 10010101?

Si considerino i simboli uno a uno. Rappresentano l'"esistenza" (se è uguale a uno) della potenza di due corrispondente. In caso contrario (=0) ne denotano l'assenza.

Spieghiamoci meglio scrivendo il numero in verticale, anzichè in orizzontale. Vale a dire, invece di 10010101, scriviamo:

1	La corrispondente potenza di due esiste
0	La corrispondente potenza di due non esiste
0	La corrispondente potenza di due non esiste
1	La corrispondente potenza di due esiste
0	La corrispondente potenza di due non esiste
1	La corrispondente potenza di due esiste
0	La corrispondente potenza di due non esiste
1	La corrispondente potenza di due esiste

Che cosa si intende con il termine "corrispondente"?

Semplicemente il "posto" che quel simbolo occupa nella numerazione all'interno del byte (insieme costituito dagli otto bit). Il primo simbolo a sinistra è il settimo, il secondo è il sesto e così via, fino al primo a destra che rappresenta il bit di "peso" zero.

Peso (pos.)	Val. (0/1)	Esiste (sì/no)	Calcolo (pot. 2)	Risult.
7	1	sì	2↑7	128
6	0	no	0	0
5	0	no	0	0
4	1	sì	2↑4	16
3	0	no	0	0
2	1	sì	2↑2	4
1	0	no	0	0
0	1	sì	2↑0	1

Il numero due (e qualsiasi altro numero) elevato alla potenza zero, fornisce come risultato "1" e non zero come potrebbe sembrare a prima vista.

Si noti, inoltre, che il "peso", cioè le posizioni, sono numerate da "7" a "0" e non da "8" a "1": anche zero è un numero!

Il numero binario 10010101 corrisponde dunque al decimale:

128+16+4+1:149

La notazione binaria è costituita da due soli simboli (0/1); quella decimale da dieci (0, 1, 2, 3, 4, 5, 6, 7, 8, 9); quella esadecimale, di cui si sente spesso parlare, da ben sedici simboli. Questi sono i "soliti" dieci della decimale, più le lettere dell'alfabeto A, B, C, D, E, F.

Per ciò che riguarda la corrispondenza tra le due notazioni, si tenga presente che per i valori tra zero e nove, la decimale e la esadecimale sono perfettamente identiche. Le cose cambiano da 10 a 15.

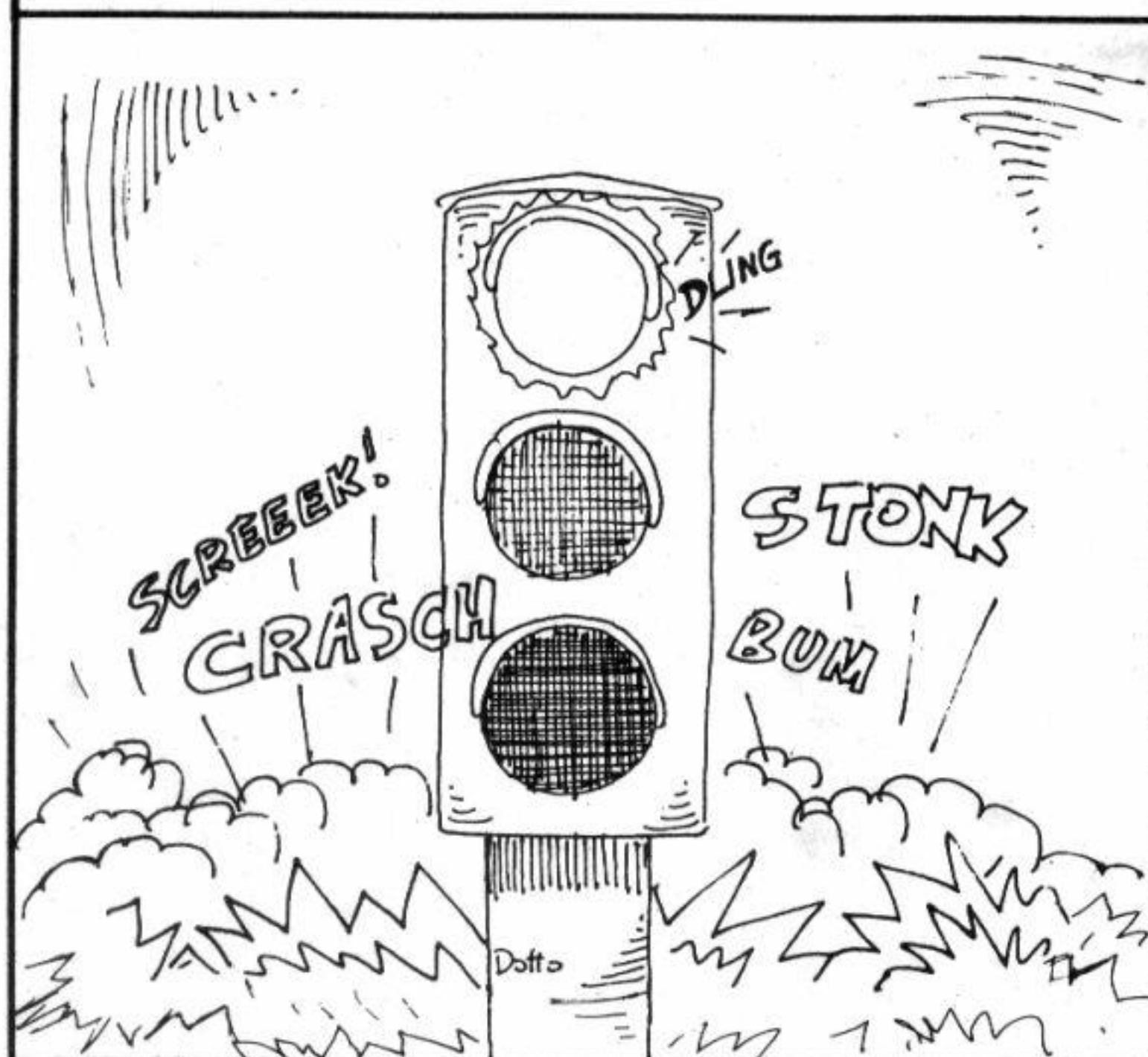
Vediamo:

Decim	*Esad.*
9	9
10	A
11	B
12	C
13	D
14	E
15	F

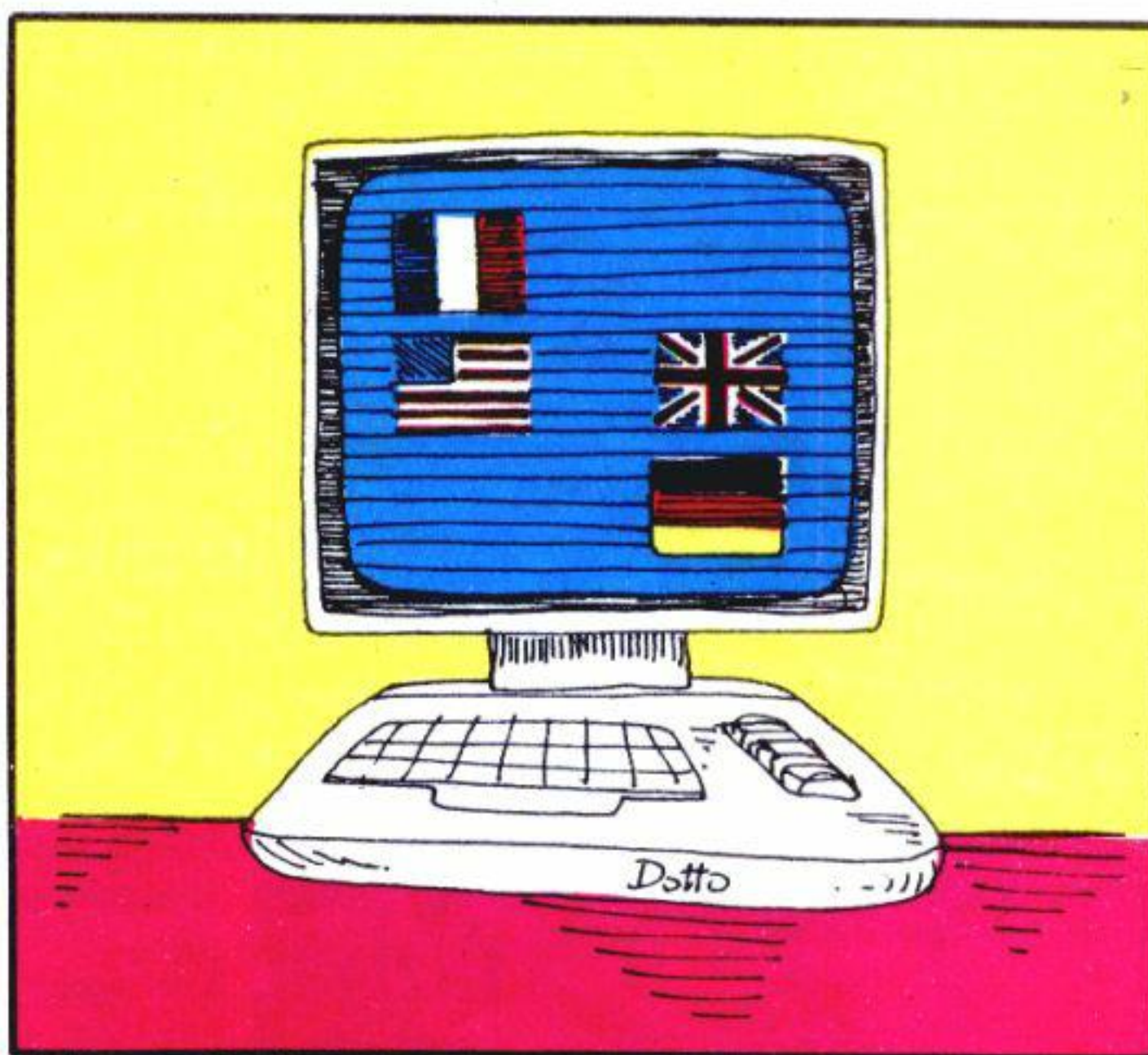
E per i numeri maggiori di 15? E' intuitivo...

Decim.	*Esad.*
16	10
17	11
...	...
31	1F
...	...
74	4A
...	...
255	FF

Per esercizio, "inventatevi" un numero compreso tra zero e 255 e "traducetelo" in binario e in esadecimale. In seguito utilizzate il programma riportato in queste pagine per controllarne la corretta conversione.

```
100 REM CONVERSIONE
110 REM DA DECIMALE
120 REM A BINARIO
130 REM ED ESADECIMALE
140 :
150 INPUT"NUMERO DECIMALE (0-255)";ND
160 X=ND:A$=""
170 IFND>255ORND<0ORND<>INT(ND)THEN150
180 FORI=0TO7:Y=X/2
190 IFY<>INT(Y)THENA$="1"+A$:GOTO210
200 A$="0"+A$
210 X=INT(Y):NEXT:PRINT"BINARIO "A$
220 X=ND/16:Y=INT(X):X=(X-Y)*16
230 PRINT"ESADECIMALE ";
240 IFY<10THENPRINTY;:GOTO260
250 PRINTCHR$(55+Y);
260 IFX<10THENPRINTX;:PRINT:RUN
270 PRINTCHR$(55+X):PRINT:RUN
```

Di che lingua sei?

Ci sono più linguaggi di programmazione o di "comunicazione"? Come scegliere il più adatto alle nostre esigenze?

di D.R. Matturro - M.L. Nitti

Dagli anni '50, periodo di nascita dei primi linguaggi evoluti, ad oggi sono stati creati tanti codici di programmazione che se na è perso il conto, senza parlare dei vari "dialetti" che crescono intorno a ognuno di questi. Per nostra fortuna non sono tutti correntemente in uso, anzi, il conto di quelli d'utilizzo comune può essere tranquillamente tenuto sulle dita delle mani.

Quali problemi, quale linguaggio?

In primo luogo ci si domanda che cosa giustifica un così spropositato fiorire di "gerghi" e, subito dopo, ci si chiede quali siano le loro differenze.

Le risposte si intrecciano tra loro, fino a diventare un'affermazione unica:

Differenti esigenze di applicazione conducono a differenti linguaggi di programmazione che si diversificano in base alle necessità che li originano.

Questo gioco di parole ci conduce a una prima constatazione:

Un linguaggio di programmazione è generalmente orientato alla soluzione di "un certo tipo" di problemi.

Per toccare con mano la cosa, torna utile inquadrare un paio di applicazioni straordinariamente differenti tra loro, quali la gestione di archivi su disco (molti dati, richiesta di precisione modesta, ricerca rapida di un dato tra migliaia) e il calcolo astronomico (pochi dati, precisione elevatissima, calcoli sofisticati, programmi relativamente brevi).

Difficilmente riuscirete a trovare un linguaggio capace di gestire con altrettanta efficacia la programmazione di software dedicato a entrambi i campi d'applicazione accennati poichè dovrebbe trattare i file in modo superlativo e nello stesso tempo avere istruzioni di calcolo estremamente complesse.

Interpreti e compilatori

Un'altra differenza sostanziale risiede nelle modalità di "traduzione" del linguaggio.

Per esser compresi anche dai lettori meno esperti, conviene aprire una parentesi esplicativa su ciò che rappresenta un linguaggio e sulle relative modalità di intervento sul calcolatore.

In primo luogo occorre precisare che esiste un solo "codice" comprensibile per il computer: il linguaggio macchina (LM). E' composto di numeri e, in particolare, di cifre in numerazione binaria. Le istruzioni di questo "originario" linguaggio sono molto "piccole" rispetto a quelli che vengono definiti linguaggi "evoluti" o ad "alto livello".

Per esempio, un singolo PRINT del Basic corrisponde ad alcune decine di istruzioni in LM, tanto è vero che le istruzioni dei linguaggi ad alto livello vengono anche definite "macroistruzioni" (dal greco: macro = grande).

Un programma "traduttore" permette di utilizzare le macroistruzioni del linguaggio evoluto poichè ne traduce il significato in adeguate "routine" di codice macchina. La differenza cui si accennava in apertura, risiede proprio nel modo in cui la "traduzione" viene effettuata.

Quando un linguaggio è interpretato

(vedi interprete Basic) la traduzione è simultanea, nel senso che viene eseguita durante l'esecuzione (RUN) del programma oppure direttamente, da tastiera. In questo caso, infatti, è possibile dare comandi diretti alla macchina.

I programmi compilatori, invece, non possono eseguire la traduzione diretta di un comando, ma devono passare attraverso la fase, appunto, di compilazione. Dapprima convertono il programma precedentemente scritto in un certo modo (che in questo caso assume il nome di "sorgente"). In seguito passa attraverso a varie fasi di compilazione.

In pratica, il programmatore deve scrivere interamente il programma prima di poterlo provare. Una volta stilato il "sorgente" lo si "compila" ottenendone una nuova versione, tradotta in LM che prende nome di "oggetto". Solo questa versione può essere eseguita dall'utilizzatore finale. E' ovvio che, se è stato commesso un errore, è necessario ripetere interamente l'operazione dato che correggerlo non è così semplice come nel Basic. Una volta messo a punto, però, il programma compilato è decisamente più veloce ed efficiente rispetto a un interprete.

Non dimentichiamo, inoltre, che esistono diversi programmi per il Commodore 64 che consentono di compilare programmi scritti in Basic. Non possono esser considerati veri e propri compilatori ma, riescono a velocizzare enormemente l'elaborazione dei listati Basic, specialmente se si lavora con le matrici.

Velocità contro praticità d'uso

Dal punto di vista dei tempi necessari alla correzione, prova e verifica dei programmi è molto più conveniente utilizzare linguaggi interpretati. Considerando la cosa dal punto di vista della velocità di esecuzione, sono invece senza dubbio più indicati i compilati, poichè vengono eseguiti direttamente in codice macchina.

Bisogna aggiungere che la maggior parte dei linguaggi esiste solo in forma compilata e sono pochi quelli interpretati, quali il Basic o il Logo oppure, forse, qualche introvabile versione di Fortran.

Le tendenze dei ricercatori sono però orientate verso formule di linguaggio molto vicine alla lingua parlata e in versione, per di più, interpretata. Linguaggi della quarta generazione come ad esempio Prolog possono esistere solo se utilizzabili in modo diretto perchè più semplici da usare soprattutto da coloro che non son pratici di informatica.

Un programma per tanti linguaggi

Presentiamo qui di seguito la versione, in diversi linguaggi, di un minigioco, allo scopo di esaminare pur se in modo superficiale, le differenze esistenti tra vari linguaggi.

Il gioco è molto semplice: si tratta di indovinare un numero. Il programma sceglie un numero tra 1 e 100, analizza le risposte di chi tira a indovinare e suggerisce se aumentare o diminuire il tiro, fino a ottenere la risposta esatta.

In queste pagine vi sono quattro procedure scritte in Pascal, in Fortran, in "C"e in Basic, anche se in realtà il linguaggio che più si presta alle realizzazioni di giochi interattivi è, come sanno gli affezionati, proprio il Basic.

Indovina un numero in Pascal

```
program indovina(input,output);
var ans: char;
    numero, mionumero:integer;
begin
ans := 's';
mionumero:=0;
repeat
  rand(numero);
  while ( mionumero <>numero) do
    begin
      writeln(' scrivi un numero');
      readln(mionumero);
      if( mionumero > numero)
      then writeln(' troppo grande');
      else writeln(' troppo piccolo');
    end
  writeln('bravo! hai indovinato');
  writeln(' vuoi giocare ancora ?');
  readln(ans);
until ( ans <> 'n')
end.
```

Si suppone che la funzione "rand" restituisca un numero a caso tra 1 e 100. E' inoltre necessario precisare che non è una funzione di biblioteca del Pascal. Il gioco continua finchè non viene risposto "n" alla domanda. Si osservi l'eleganza della strutturazione logica del Pascal.

Abbiamo usato una versione non troppo "sofisticata" di Fortran, per mostrare una strutturazione diversa del programma. La subroutine "rand1" restituisce numeri a caso con distribuzione uniforme tra 0 e 1. Non è una funzione di libreria. Si può osservare come la ripetizione del gioco a seconda della risposta debba necessariamente avvenire con un "salto".

Indovina un numero in Fortran

```
C     indovina un numero tra 1 e 100
      program indovina
      character ans
100   call rand1(x)
      numero = x*100
200   print*,' dimmi un numero'
      read*,mionum
      if (mionum.lt.numero) then
        print*,'troppo basso'
      else
if (mionum.eq.numero) then
      print*,'bravo, hai indovinato!'
      print*,'vuoi giocare ancora?'
      read (*,300) ans
      if(ans.eq.'s')goto100
      stop
endif
print*,'troppo alto'
      endif
      goto200
300-format(a1)
     end
```

Indovina un numero in C

```
   include <stdio.h>
main()
{{
ma
   int numero, mionumero;
   char ans;
   ans = 's';
   mionumero = 0;
   while ( ans == 's')
{{
whinumero = rand();
   while (numero != mionumero)
   {{
while printf("//N scrivi un numero //n");
      scanf("%d",&mionumero);
      if(mionumero > numero)
      printf("//N diminuisci");
      else
      printf("//n aumenta");
   }}
   printf("//N bravo! hai indovinato");
   printf("//N vuoi giocare ancora ? ");
   scanf("%c",&ans);
   mionumero t0;
   }}
}}
```

Ecco per ultimo il listato in Basic, sul quale non c'è molto da dire.

Indovina un numero in BASIC

```
 10 MN=0
 20 N=INT(RND(0)*100)
 30 INPUT "DIMMI UN NUMERO";MN
 40 IF MN>N THEN PRINT "TROPPO GRANDE":GOTO30
 50 IF MN<N THEN PRINT "TROPPO PICCOLO":GOTO30
 60 PRINT "BRAVO, HAI INDOVINATO!"
 70 INPUT "VUOI GIOCARE ANCORA";AN$
 80 IF AN$="S" THEN 10
 90 IF AN$T"N" THEN END
100 PRINT "NON HO CAPITO"
110 GOTO 70
```

Teo Rusconi ha appena sfatato la leggenda secondo la quale i floppy disc sono tutti uguali

Difatti sembrano tutti uguali finchè non si osserva con attenzione il jacket. Qui termina l'uguaglianza.

La maggior parte delle società costruttrici sigillano i dischi un punto qui, un punto là, lasciando parte dei lembi non sigillati.

Prima o poi ai lembi accadono cose naturalissime: si gonfiano, si curvano, si raggrinziscono... in poche parole si aprono.

GLI ALTRI DISCHETTI
chiusi un punto qui, un punto là lasciano gran parte dei lembi aperti.

DISCHETTI MEMOREX
con lembi completamente saldati su tutta la superficie.

Con penne, matite, unghie persino un ragazzino di quattro anni come Teo può infilarsi in quegli spazi aperti.

Naturalmente è un danno enorme perchè se si inserisce qualcosa di molle e slabbrato nel disc-drive quest'ultimo può incepparsi; si può rovinare la testina e si possono perdere i dati. Questo può accadere con gli abituali sistemi di chiusura ma non con i dischetti Memorex che usa un procedimento esclusivo chiamato "Solid-Seam Bonding".

Con questo sistema ogni singolo millimetro quadrato dei lembi di tutti i dischi Memorex viene sigillato ermeticamente, rendendoli più rigidi e più resistenti.

È un sistema che consente al floppy disc di sostenere ogni assalto, che impedisce alla testina di rovinarsi e ai dati di andare perduti.

Il che sta a dimostrare che un floppy disc Memorex non è uguale a tutti gli altri: è migliore. E il sistema di saldatura è solo un esempio della cura infinita con cui viene prodotto ogni floppy disc Memorex; sia esso da 8", da 5 1/4" o il nuovo 3 1/2". Questa estrema accuratezza dà la garanzia che ogni disco Memorex è al 100% perfetto.

La prossima volta che acquistate un floppy disc - o qualche centinaio - ricordate: non tutti i dischetti sono uguali... Memorex vi mette al riparo da qualsiasi inconveniente.

MEMOREX

è importante scegli

MEMOREX

A Burroughs Company

BURROUGHS-MEMOREX S.P.A.
Divisione Computer Media
Via Ciro Menotti, 14 Tel. 02/718551
20129 MILANO MI

Turbotape sonoro

Digitate questo semplice programma e il vostro registratore non sarà più... muto!

di Carlo e Lorenzo Barazzetta

Il turbotape è un programma di utilità quasi indispensabile per ogni possessore di Commodore 64 perchè permette di salvare, caricare e verificare programmi su nastro in modo assai più veloce che con i normali comandi SAVE, LOAD, VERIFY risparmiando tempo e nastro.

Questo programma è piuttosto diffuso e la rivista su nastro della Systems Editoriale (Software Club) ne fa spesso uso per registrare i programmi.

E' possibile renderlo ancora più versatile e noi di Commodore Computer Club ci siamo permessi di aggiungere, ai comandi già disponibili (<—S, <—L, <—V), un nuovo comando che consente di "ascoltare" il rumore dei programmi durante la fase di caricamento degli stessi.

E' ovvio che, prima di digitare il listato, è necessario caricare e attivare il turbotape perchè il nuovo comando (che si impartisce con <—P) non funziona senza la presenza degli altri tre.

Dopo aver digitato il listato pubblicato su queste pagine, potrete salvarlo anche con <—S e, per utilizzarlo in seguito, ricaricarlo col solito <—L e abilitare il comando <—P con RUN.

Come usare il comando <—P

Dopo aver digitato il tasto di freccia a sinistra, il carattere "P" e battuto il tasto Return, appare sullo schermo il consueto messaggio: PRESS PLAY ON TAPE.

A questo punto, premendo il tasto PLAY del registratore, lo schermo diventerà del colore del bordo e potrete sentire dall'audio del vostro apparecchio Tv il rumore dei programmi salvati sulla cassetta.

Per ritornare al Basic è sufficiente premere i tasti Run/Stop e Restore.

A che serve il nuovo comando?

Il comando <—P serve, come già detto, per sentire i suoni prodotti dai programmi registrati su cassetta. Quando, infatti, un programma viene registrato tramite <—S oppure SAVE, viene dapprima registrata una parte iniziale con il nome del programma e altri segnali per il sincronismo e, successivamente, i dati del programma stesso.

Ascoltando la parte iniziale, sentiremo un sibilo nitido mentre, quando il nastro passerà sulla sezione dati, si udrà un rumore confuso.

Il nuovo comando, quindi, si rivela utile per la ricerca dei programmi salvati su cassetta qualora non si sapesse il punto preciso in cui questi iniziano.

Facendo scorrere il nastro con i tasti Rewind e F.FWD e premendo poi il tasto Play potremo capire dal suono se siamo capitati sulla parte iniziale di un programma (sibilo) oppure no (rumore confuso).

Nel primo caso dobbiamo premere il tasto Stop del registratore, i tasti Run/stoptrestore e battere il comando <—L per caricare il programma. Nel secondo caso, al contrario, dobbiamo continuare la ricerca. Questa funzione potrebbe, con minore comodità, essere svolta da un normale registratore dotato di altoparlante (sempre che lo si abbia a portata di mano).

Inoltre per quei programmi con problemi di caricamento si può agire nel modo seguente: una volta riscontrato il fatidico messaggio di LOAD ERROR si riavvolgerà la cassetta all'inizio del programma e si batterà il comando <—P per sentire il sibilo iniziale.

Se noteremo che questo non è molto nitido, agiremo con un cacciavite sulla vite di regolazione dell'azimuth finchè il sibilo sarà perfettamente pulito. In seguito potremo premere Run/stop e Restore e il comando <—L per caricare normalmente il programma.

Probabilmente dovrete fare queste operazioni più volte perchè il sibilo dura solo un paio di giri e non riuscirete quasi mai a "sintonizzarvi" bene al primo colpo.

Infine il comando <—P può essere usato per sapere se un programma è stato salvato con SAVE (normale sintassi Commodore) oppure con <—S (sintassi T/T). Infatti il sibilo di un programma salvato con SAVE è meno acuto di uno salvato con <—S e il rumore del programma è meno confuso.

```
100 REM  COMMODORE 64
102 REM  E TURBO TAPE
104 :
110 REM  COL COMANDO ←P
112 REM  SI ASCOLTA
114 :
120 REM  IL REGISTRATORE
124 :
130 REM  DI LORENZO BARAZZETTA
140 :
150 FOR A=0 TO 22:READ B
160 IF B>=0 AND B<256 THEN 180
170 PRINT"ERRORE NEI DATI";:END
180 C=C+B:POKE 50096+A,B:NEXT
190 IF C<>2710 THEN 170
200 P=50038
210 POKE P,76:POKE P+1,176:POKE
     P+2,195
220 :
230 DATA  201,80,240,3,76,8,175
     ,32,23
240 DATA  248,32,125,196,173,13
     ,220,74
250 DATA  141,24,212,24,144,246
     ,-1
260 END
```

Viaggio nelle "adventure"

Immaginate di essere Indiana Jones e di trovarvi di fronte a una muta di lupi affamati mentre la proverbiale principessa invoca il vostro aiuto.

di Michele Maggi

Ci siamo recentemente occupati di uno dei più famosi adventure, Zork.

Illustreremo questa volta la spiegazione in dettaglio del funzionamento di un adventure, in modo da mettervi in grado di costruirne uno tutto vostro.

Il primo passo

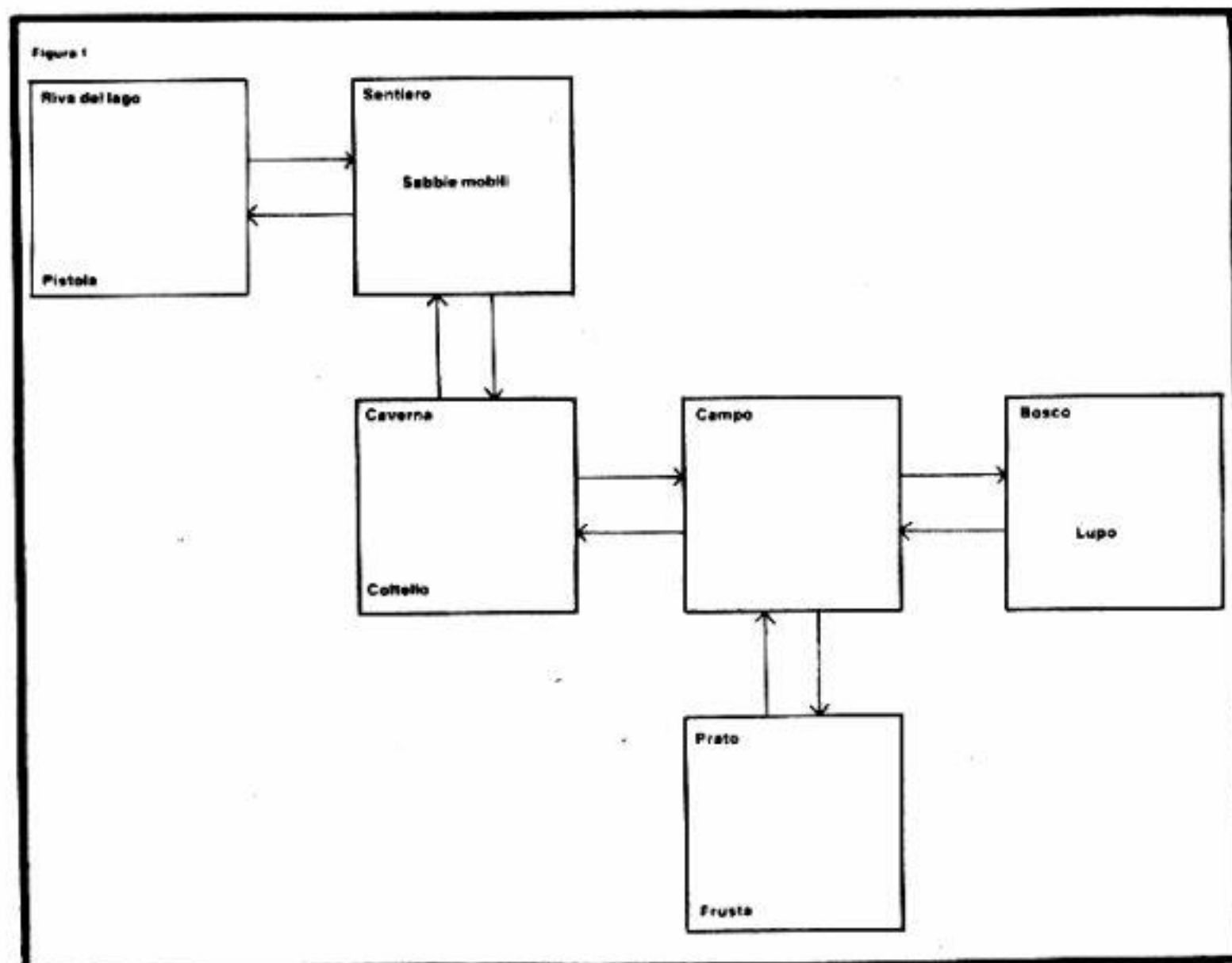
Il primo passo verso la costruzione di un adventure è inventare una trama e stabilire il compito del protagonista all'interno del gioco.

Per ragioni di spazio e, soprattutto, di chiarezza, nell'adventure pubblicato qui di seguito sono stati evitati di proposito una trama (sceneggiatura) intricata o un compito difficile da portare a termine.

Una volta inventata la trama e stabilito lo scopo dell'adventure, è la volta della descrizione dei luoghi da visitare e questa, sicuramente, è la parte più avvincente del gioco.

Non c'è limite al numero di locazioni che possiamo inserire, ma, per il momento, converrà mantenerlo piuttosto basso per essere sicuri di comprendere a fondo il funzionamento di un programma-tipo.

In primo luogo bisogna disegnare un bozzetto raffigurante la disposizione delle locazioni (che d'ora in poi chiameremo convenzionalmente stanze), come in figura 1.

Successivamente dovremo inserire la mappa scelta in una matrice di 5 quadrati per 5, come in figura 2.

Solo tramite questa operazione saremo in grado di muoverci nell'adventure.

Subito dopo numereremo i 25 quadrati in modo che ogni stanza sia individuata da un numero ben preciso.

Esempio:
Riva del fiume = 1
Sentiero = 2
Caverna = 7
Campo = 8
Bosco = 9
Prato = 13

Resta sottinteso che i numeri dei quadrati inutilizzati non verranno cosiderati nel corso del gioco.

Quattro passi per l'adventure

La formula che ci permetterà di spostarci di stanza in stanza è di una semplicità estrema.

Assumendo che il punto di partenza sia la caverna (stanza n. 7), per andare a Nord dovremo sottrarre al numero della nostra stanza il numero dei quadrati che formano un lato della matrice, cioè:

7-5=2

Il numero 2, infatti, è il numero della stanza posto immediatamente al di sopra (Nord) della caverna, cioè il sentiero.

Analogamente, sommeremo, invece di sottrarre, per andare verso Sud. In tal modo se, nel corso del gioco, ci troviamo nel campo (stanza 8); sommando 8+5 otteniamo 13, cioè il numero di stanza corrispondente al prato.

Per andare a Est oppure a Ovest basterà semplicemente aggiungere o sottrarre uno, qualunque siano le dimensioni della matrice.

Per esempio, se ci troviamo nella caverna (stanza 7) e vogliamo andare a Est, la formula sarà la seguente:

7+1=8

cioè il campo.

Semplice, vero?

Quanto detto finora vale per una matrice di 5x5, ma nessuno ci impedisce di aumentare le dimensioni dell'adventure per esempio a 10x10, purchè nella formula venga inserito il valore 10 e non più 5.

Gli oggetti

In ogni adventure esistono vari oggetti, soggetti a spostamenti, che danno agli adventure quel realismo che li caratterizza. In sostanza saremo in grado di prendere un oggetto in una stanza, deporlo in un'altra ed essere sicuri di ritrovarlo successivamente nello stesso punto in cui l'abbiamo lasciato. Il programma, in altre parole, "deve" ricordare il luogo in cui è stato abbandonato un qualsiasi oggetto.

Dobbiamo, inoltre, avere il diritto di chiedere al programma, in qualsiasi momento, di informarci sulla nostra dotazione che si modifica di volta in volta, a seconda se abbandoniamo o prendiamo gli stessi oggetti.

I pericoli

Tutti gli adventures sono sempre stracolmi di pericoli, mortali o meno, ma sempre indesiderati.

Si scamperà da alcuni solo se in possesso di determinati oggetti.

Nel gioco proposto, ad esempio, sopravviverete alle sabbie mobili solo se in possesso della frusta che ci permetterà di aggrapparci a un albero e potremo uccidere il lupo solo sparandogli con la pistola presa in precedenza, altrimenti...

I comandi

Durante lo svolgimento del gioco verrà continuamente richiesto il da farsi e potremo rispondere indicando una direzione (N, S, E, O) o impartendo un comando vero e proprio relativo ad azioni più complesse come, ad esempio, prendere, gettare, uccidere e così via.

In riquadro a parte sono elencate tutte le parole chiave relative al gioco pubblicato, "Caccia al Lupo".

E' possibile insegnare al computer a distinguere, in una frase-comando, il verbo dal nome, assumendo che il primo comando sia il verbo e la parola successiva all'eventuale articolo, il nome (esempio: prendere la frusta).

Prima di passare ad analizzare dettagliatamente il programma ci soffermeremo sulla trama proposta e, soprattutto, sulla strategia vincente.

Caccia al lupo

Il listato, come già detto, è un microadventure senza nessuna pretesa, a parte quella di insegnare come si può realizzare un gioco di questo tipo.

La trama

La trama non può che essere banale, data la brevità del listato e quindi non c'è posto per vistosi colpi di scena.

Vi trovate in un luogo desolato e avete il compito di trovare e uccidere un enorme lupo che in passato ha fatto strage di uomini e animali.

Avete chiaramente bisogno di armi, ma quale arma sarà in grado di uccidere un lupo così grosso?

Muovendovi molto cautamente dovete cercare la pistola che, purtroppo, avete perso per strada durante il viaggio di andata.

Una volta trovata la pistola, potrete finalmente (se siete ancora vivi) uccidere il lupo.

La strategia vincente

Come detto prima, in questo adventure la strategia vincente è una sola.

Dal punto di partenza, la caverna, vi muovete verso Est, quindi verso Sud e qui, nel prato, trovate una frusta che dovrete prendere (sempre che vogliate sopravvivere, s'intende).

Poi tornate alla caverna e andate a Nord, ma... cadete nelle sabbie mobili!

Cosa fare? Cercate di uscire e, come per incanto, con un'impresa degna di Indiana Jones vi troverete fuori dalle sabbie mobili.

Da qui ora andate a Ovest e arrivate in riva a un fiume e... sorpresa !! E' proprio qui che avete perduto la vostra pistola. Prendetela senza esitare e andate a cercare il lupo perchè ora non avete più nulla da temere.

Potete in qualsiasi momento chiedere aiuto semplicemente rispondendo "A" alla richiesta di istruzioni.

E' interessante notare come i consigli cambino in funzione della stanza in cui ci si trova.

Provate, ad esempio, a chiedere aiuto quando siete nelle sabbie mobili o quando siete di fronte al lupo.

Un'ultima osservazione prima di analizzare il programma.

Così come viene pubblicato, l'adventure si presenta piuttosto scarno e ciò è dovuto alle descrizioni che fatalmente, per questioni di spazio, sono state ridotte al minimo. Ciò non toglie, però, che nei vostri adventure possiate, anzi dobbiate inserire descrizioni molto più esaurienti e coinvolgenti.

Il genere a cui si ispira l'adventure può essere quantomai vario: fantascientifico, western, horror, giallo, thriller. Non ci sono limiti se non quelli posti dalla vostra fantasia.

Come funziona il programma

10-40 inizializzano le variabili relative al numero (locazione) della stanza, ai nomi degli oggetti che vi si troveranno e a una piccola descrizione degli stessi.

50 setta la posizione iniziale nella stanza (N.7). E' possibile cambiare a piacimento il valore di questa variabile; anzi, provate a cambiarlo in 13 e vedrete che, invece di partire dalla caverna, partirete dal prato. I più bravi potranno inserire una routine che randomizzi questo valore tutte le volte che inizia il gioco.

60 assegna a P il valore di P2 che rappresenta la nuova posizione.

70-90, in funzione del valore di P, mandano alla riga che ne contiene la descrizione corrispondente.

Gli zeri hanno semplicemente lo scopo di far selezionare l'esatto numero di linea (vedere in dettaglio sul manuale del vostro computer l'uso dell'istruzione ON...GOTO).

100-110 stampano la descrizione dell'eventuale oggetto posto nella stessa stanza in cui si trova il giocatore.

120 e 130 controllano se il giocatore si trova in una delle stanze contenenti i pericoli e descrivono il pericolo in questione.

140-180 stampano le possibili uscite.

190 richiede le istruzioni.

200-210 hanno la stessa funzione delle linee 120-130, ma per impedire i movimenti.

220 controlla se la risposta è una direzione.

240-270 spostano il giocatore facendo uso della formula descritta nell'articolo.

280 impedisce al giocatore di andare in una direzione non consentita,

290 impedisce di muoversi in presenza di pericolo.

300 manda alla routine di stampa degli oggetti in possesso del giocatore.

310 controlla se il giocatore è nelle sabbie mobili e se vuole uscire.

320 predispone il messaggio di aiuto.

330-420 separano il verbo dal nome e lo confrontano con le parole chiave.

430-450 stampano la dotazione.

480-500 stampano il messaggio di aiuto in funzione della posizione del giocatore.

510-590 controllano il possesso o meno degli oggetti e permettono di prenderli o di lasciarli.

600 rimanda alla stampa delle direzioni possibili.

610-620 controllano se il giocatore ha la frusta nel caso in cui sia capitato nelle sabbie mobili.

640-670 controllano se il giocatore può uccidere il lupo.

680-790 descrivono le stanze e ne stabiliscono le uscite.

800-920 routines di morte e di richiesta di una nuova partita.

Lista variabili

SM = sabbie mobili, è settata a 2 quando il pericolo è scampato.

LU = lupo, funziona come sopra.

OG(I) = locazioni che inizialmente contengono gli oggetti.

OG$(I) = nomi degli oggetti.

OD$(I) = descrizioni degli oggetti.

P = posizione.

P2 = posizione successiva.

N = contiene il numero di stanza che eventualmente si trova a Nord della posizione.

S = come sopra ma per Sud.

E = come sopra ma per Est.

O = come sopra ma per Ovest.

I$ = struzione.

V$ = verbo.

N$ = nome.

D = dotazione.

A = aiuto.

△

Parole chiave e comandi

N = Nord.
S = Sud.
E = Est.
O = OvEst.

△

Abbandonare
Afferrare
Ammazzare
Buttare
Gettare
Lasciare
Prendere
Sparare
Uccidere
Uscire

△

D = Dotazione
A = Aiuto

△

N.B. Per ciò che riguarda i verbi, il programma è predisposto per accettarli solo all'infinito, come se fosse sottinteso "Io voglio".

E' assunto che quando si è davanti al lupo se si scrive "uccidere" o "sparare", sia sottinteso "al lupo". Perciò una frase tipo "uccidere il lupo" avrà lo stesso effetto di "uccidere il mostro" o di "uccidere il programmatore".

Il risultato sarà comunque l'uccisione del lupo, purchè siate in possesso della pistola.

△

```
  1 REM  CACCIA AL LUPO
  2 :
  3 REM  ADVENTURE PER
  4 REM  QUALSIASI COMMODORE
  5 :
  6 REM  DI MICHELE MAGGI
  7 :
 10 PRINTCHR$(147):SM=0:LU=0:FO
    R I=1 TO 3:READ OG(I),OG$(I
    ),OD$(I):NEXT
 20 DATA  7,IL COLTELLO,PER TER
    RA C'E' UN COLTELLO
 30 DATA  1,LA PISTOLA,QUI C'E'
     LA TUA PISTOLA
 40 DATA 13,LA FRUSTA,A TERRA C
    'E' UNA LUNGA FRUSTA
 50 P=7:GOTO 70
 60 P=P2
 70 IF P<11 THEN ON PGOTO 680,7
    00,0,0,0,0,720,740,780,0
 80 IF P<21 THEN ON P-10GOTO 0,
    0,760,0,0,0,0,0,0,0
 90 IF P<26 THEN ON P-20GOTO 0,
    0,0,0,0
100 FOR I=1 TO 3:IF OG(I)=P THE
    N PRINTOD$(I)
110 NEXT
120 IF P=2 AND SM<>2 THEN 800
130 IF P=9 AND LU<>2 THEN 810
140 PRINTCHR$(17)"PUOI ANDARE"C
    HR$(17)
150 IF N>0 THEN PRINT" A NORD"
160 IF S>0 THEN PRINT" A SUD"
170 IF E>0 THEN PRINT" A EST"
180 IF O>0 THEN PRINT" A OVEST"
190 I$="":INPUT "[2 DOWN]ISTRUZ
    IONI";I$:PRINT"[CLEAR]"
200 IF P=2 AND SM<>2 THEN 290
210 IF P=9 AND LU<>2 THEN 290
220 IF I$="N" OR I$="S" OR I$="
    E" OR I$="O" THEN 240
230 GOTO 300
240 IF I$="N" AND N>0 THEN P2=P
    -5:GOTO 60
250 IF I$="S" AND S>0 THEN P2=P
    +5:GOTO 60
260 IF I$="E" AND E>0 THEN P2=P
    +1:GOTO 60
270 IF I$="O" AND O>0 THEN P2=P
    -1:GOTO 60
280 PRINTCHR$(17)"NON PUOI ANDA
    RE A ";I$:GOTO 70
290 IF I$="N" OR I$="S" OR I$="
    E" OR I$="O" THEN PRINT"NON
     PUOI FARLO ADESSO !!!":GOT
    O 70
300 IF I$="D" THEN 430
310 IF I$="USCIRE" AND P=2 AND
    SM<>2 THEN 610
320 IF I$="A" THEN 480
330 FOR I=1 TO LEN(I$):IF MID$(
    I$,I,1)=" " THEN 360
340 NEXT
350 PRINT"USARE PIU' PAROLE.":G
    OTO 70
360 FOR SX=1 TO LEN(I$): IF MID
    $(I$,SX,1)=" " THEN 380
370 NEXT
380 V$=LEFT$(I$,SX-1):N$=RIGHT$
    (I$,(LEN(I$)-SX))
390 IF V$="PRENDERE" OR V$="AFF
    ERRARE" THEN 510
400 IF V$="GETTARE" OR V$="BUTT
    ARE" OR V$="LASCIARE" OR V$
    ="ABBANDONARE" THEN 560
410 IF V$="UCCIDERE" OR V$="AMM
    AZZARE" OR V$="SPARARE" THE
    N 630
420 PRINT"[DOWN]NON SO COSA VUO
    L DIRE [RVS]";V$:GOTO 70
430 PRINT"[CLEAR]LA TUA DOTAZIO
    NE E'[DOWN]":D=0
440 FOR I=1 TO 3:IF OG(I)=-1 TH
    EN PRINTOG$(I):D=D+1
450 NEXT
460 IF D=0 THEN PRINT"NULLA."
470 GOTO 70
480 IF  P=9  THEN PRINT"PROVA A
    D UCCIDERLO....":GOTO 70
490 IF P=2 THEN PRINT"PROVA AD
    USCIRE...":GOTO 70
500 PRINT"NON HAI BISOGNO DI AI
    UTO QUI.":GOTO 70
510 FOR I=1 TO 3:IF OG$(I)=N$ T
    HEN 530
520 NEXT
530 IF OG(I)=-1 THEN PRINT"CE L
    'HAI GIA'.":GOTO 70
```

```
540 IF OG(I)<>P THEN PRINT"QUI
    NON C'E' ";N$:GOTO 70
550 PRINT"VA BENE, HAI PRESO "O
    G$(I):OG(I)=-1:GOTO 70
560 FOR I=1 TO 3:IF OG$(I)=N$ T
    HEN 580
570 NEXT
580 IF OG(I)<>-1 THEN PRINT"NON
     L'HAI.":GOTO 70
590 PRINT"VA BENE, HAI GETTATO
    "OG$(I):OG(I)=P:GOTO 70
600 GOTO 140
610 IF OG(3)=-1 THEN PRINT"OK,
    CON LA FRUSTA TI AGGRAPPI A
     UN RAMO.":SM=2:GOTO 70
620 PRINT"NON HAI NULLA A CUI A
    GGRAPPARTI...":GOTO 900
630 IF P<>9 THEN 670
640 IF OG(2)=-1 THEN PRINT"OK,
    L'HAI UCCISO !! HAI VINTO L
    A SFIDA.":GOTO 850
650 IF OG(1)=-1  THEN PRINT"CON
     QUEL TEMPERINO ? VUOI SCHE
    RZARE !!":GOTO 820
660 PRINT"NON HAI NULLA CON CUI
     UCCIDERLO":GOTO 820
670 PRINT"QUI NON C'E' NESSUNO
    DA UCCIDERE.":GOTO 70
680 PRINT"[2 DOWN]SEI IN RIVA A
    D UN FIUME"
690 N=0:E=2:S=0:O=0:GOTO 100
700 PRINT"[2 DOWN]SEI SU UN SEN
    TIERO FANGOSO"
710 N=0:E=0:S=7:O=1:GOTO 100
720 PRINT"[2 DOWN]SEI IN UNA CA
    VERNA CON DUE USCITE"
730 N=2:E=8:S=0:O=0:GOTO 100
740 PRINT"[2 DOWN]SEI IN UN CAM
    PO DELIMITATO DA UN BOSCO"
750 N=0:E=9:S=13:O=7:GOTO 100
760 PRINT"[2 DOWN]SEI IN UN PRA
    TO"
770 N=8:E=0:S=0:O=0:GOTO 100
780 PRINT"[2 DOWN]SEI VICINO A
    UN BOSCO"
790 N=0:E=0:S=0:O=8:GOTO 100
800 PRINT"SEI CADUTO NELLE SABB
    IE MOBILI":GOTO 190
810 PRINT"UN ENORME LUPO ESCE D
    AL BOSCO.":GOTO 190
820 PRINT"[DOWN]IL LUPO AVANZA
    LENTAMENTE VERSO DI TE,"
830 PRINT"TU CERCHI DI SCAPPARE
     MA INUTILMENTE,"
840 PRINT"IL LUPO TI RAGGIUNGE
    E TI DIVORA...":PRINT"[2 DO
    WN]         GNAM GNAM"
850 FOR T=1 TO 3000:NEXT
860 PRINT"[DOWN]    GIOCHI ANCO
    RA (S/N)"
870 GET A$:IF A$="" THEN 870
880 IF A$="S" THEN  RUN
890 END
900 PRINT"IL FANGO TI ATTIRA SE
    MPRE PIU' GIU',"
910 PRINT"TI RIEMPIE LA BOCCA E
     TU NON PUOI PIU'"
920 PRINT"RESPIRARE ... MI DISP
    IACE.":GOTO 850
```

Una stampante tuttofare

La stampante a colori Okimate 20 è stata provata simulando "ambienti" d'uso tra i più disparati. I risultati sono convincenti: eccoli.

di Alessandro de Simone

A vederla così piccola e leggera (cm 33x19x6 e peso nettamente inferiore ai 3 Kg), viene subito in mente di classificare Okimate 20 tra i prodotti di utilizzo esclusivamente amatoriale.

Imparando a usarla, al contrario, non si può fare a meno di immaginarne l'uso in tutte quelle occasioni in cui la "solita" stampante non soddisfa pienamente. Ma, bando alle chiacchiere, esaminiamo i lati positivi del prodotto.

- Collegabile direttamente (senza, cioè, interfaccia di alcun tipo) al vostro Commodore (Vic 20, C-16, Plus/4, C-64, C-128) mediante il solo cavo seriale.
- Totalmente compatibile con il "protocollo" standard Commodore (vedi, infatti, il listato dimostrativo di queste pagine e il corrispondente Output).
- In unione col word processor "Easy Sript" (con cui è stato realizzato il presente articolo) funziona perfettamente.
- In unione con programmi "difficili" (come il Print Shop) funziona in modo ineccepibile, dimostrando che è possibile "scaricare" pagine in alta risoluzione, realizzate nei modi più diversi, senza alcun problema.
- Il dischetto presente nella confezione consente di riportare su stampante (in 16 colori oppure in cinque tonalità, dal bianco al nero) disegni eseguiti con uno dei seguenti programmi grafici, ben noti alla maggior parte dei nostri lettori: Doodle, Koala, Edumate, Super sketch, Flexidraw, Paint magic, Calkboard, Sorcerer's Apprentice, Blazing paddles. Un menu permette la scelta, la ricerca e la stampa del file-disegno.
- Massima silenziosità dovuta alla testina termica di cui è dotata la stampante.
- Possibilità d'uso di carta normale, carta con superficie speciale lucida (per le migliori prestazioni), in rulli o in pacchi perforati, carta termica (da usare rimuovendo il nastro, che risulta in questo caso del tutto inutile), carta lucida (per disegni tecnici) e, grande novità, supporto in acetato, da utilizzare con le lavagne luminose.
- Il nastro, nero o a colori, garantisce una qualità di riproduzione molto buona (validissima per ottenere fotocopie di documenti di qualità e nelle protezioni con lavagne luminose) perchè è del tipo che si utilizza una sola volta, si-

mile a quelli in dotazione a macchine da scrivere di qualità elevata. Durata del nastro: 120mila caratteri (b/n), 35mila a colori. La stampante, usata in b/n con carta termica, non richiede, ovviamente, l'inserimento della cartuccia inchiostrata.

- Velocità accettabile (40, 60, 80 c.p.s. a seconda del modo d'uso).
- Matrice 14x14 oppure 7x14 (alta qualità oppure alta velocità).
- Tre tipi di stampa (Pica, Elite, Condensato).
- Riproduzione di formule matematiche, anche con potenze e indici (vedi listato).
- Programmabilità completa dei caratteri definiti dall'utente.

A chi può servire

La stampante a colori Okimate 20 è, anzitutto, una stampante "comoda". Troppo spesso sono state viste in giro stampanti a colori (e anche b/n) che, pur se in grado (in teoria) di riportare videate in alta risoluzione, in pratica abbandonavano l'utente al suo destino, confortandolo solo con un paio di pagine di istruzioni (poco chiare) su come realizzare le hard copy.

La Okimate 20, invece, può essere usata da un principiante che non sa nulla di programmazione: gli basterà caricare il dischetto in dotazione e, una volta lanciato il programma, inserire il proprio dischetto su cui sono memorizzati i file-disegni che intende riprodurre.

A parte la soddisfazione di vedere su carta (anche a colori!) i disegni realizzati con tanta fatica, non dobbiamo trascurare la possibilità di effettuare copie su supporto di acetato, necessario per utilizzare le lavagne luminose.

Indispensabile, proprio per quest'ultima opportunità, anche in scuole e aziende, la Okimate 20 può sostituire il plotter nei casi in cui non sia disponibile del software in grado di "scaricare" schermate realizzate in alta risoluzione.

Per ulteriori informazioni i lettori sono pregati di rivolgersi direttamente all'importatore:

Technitron
Via Milanofiori, Pal. E/2
20094 Assago (Mi)
Tel. 02/824 2112

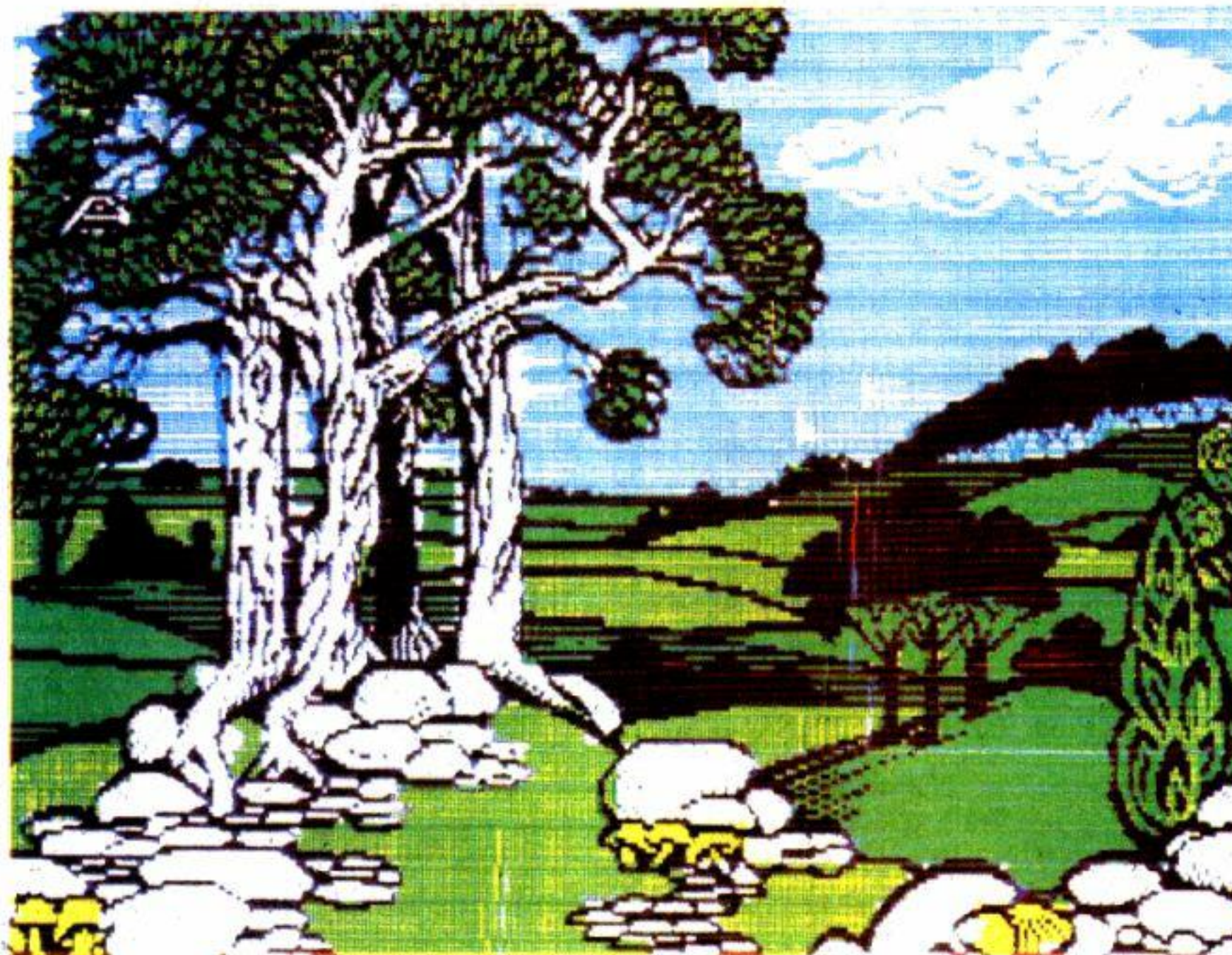



FIGURA 1

```
100 rem programma dimostrativo che evidenzia
110 rem le caratteristiche della stampante
120 rem    o k i m a t e  -  2 0
130 :
140 open 1,4,7
150 rem 1= n.file
160 rem 4= n.periferica (stampante)
170 rem 7= can.second. (maiuscolo/minuscolo)
180 r$=chr$(13):rem rigo vuoto
190 print#1,"PROVA SULLA STAMPANTE A COLORI OKIMATE-20 per sistemi COMMODORE"
200 fori=1to65:print#1,"-";:next:print#1:print#1
210 print#1,"Questa e' una stampa normale"r$
220 print#1,chr$(14)"chr$(14) raddoppia i CARATTERI"r$
230 print#1,chr$(15)"chr$(15) ritorna alla larghezza normale"r$
240 print#1,chr$(18)" chr$(18) attiva il REVERSE MODE "r$
250 print#1,chr$(145)"chr$(145) attiva maiuscolo e grafico:ABCDE=abcde"r$
260 print#1,"chr$(27) corrisponde al comado:ESC"r$
270 print#1,chr$(27)chr$(67)"ESC+67 e' il comando della sottolineatura"r$
280 print#1,chr$(27)chr$(68)"ESC+68 e' il comando che la interrompe"r$
290 print#1,"Esempio si formula matematica"r$
300 print#1,"X con 7 elev.alla decima ";
310 print#1,"piu' X con 4 per Y elevato a (";
320 print#1,"H piu' R) meno X "
330 print#1,"elevato alla J piu' Y:"r$
340 giu$=chr$(27)+chr$(76):su$=chr$(27)+chr$(74)
350 print#1,"X"giu$"7"su$;su$"10"giu$;
360 print#1,"+ X"giu$"4"su$" * Y"su$"(H+R)"giu$"-X"su$"(J+Y)"r$
370 print#1,giu$"Ritorno alla stampa normale"r$
380 print#1,"Caratteri speciali per la stampa: chr$(X)"r$
390 print#1,chr$(30)"CHR$(30) per scrivere con caratteri PICA"r$
400 print#1,chr$(14)"(30+14) caratteri PICA allargati"r$
410 print#1,chr$(28)"CHR$(28) per scrivere in ELITE (17 c.p.p.)"r$
420 print#1,chr$(14)"(28+14) caratteri  ELITE allargati"r$
430 print#1,chr$(29)"CHR$(29) per scrivere CONDENSATO (12 c.p.p.)"r$
440 print#1,chr$(14)"(29+14) caratteri in CONDENSATO allargati"r$
450 print#1,chr$(15)"Con CHR$(15) si ritorna la modo di stampa PICA"r$
460 print#1,"Esempi di caratteri semigrafici COMMODORE:"
470 print#1,chr$(145)"alfabeto:aA bB cC dD eE fF gG hH iI jJ kK lL"
480 print#1,chr$(145)"mM nN oO pP qQ rR sS tT uU vV wW xX yY zZ"
490 close1

ready.
```

SCALA: 10 cm.

FIGURA 2

PROVA SULLA STAMPANTE A COLORI OKIMATE-20 per sistemi COMMODORE

Questa e' una stampa normale

chr$(14) raddoppia i CARATTERI

chr$(15) ritorna alla larghezza normale

chr$(18) attiva il REVERSE MODE

CHR$(145) ATTIVA MAIUSCOLO E GRAFICO:♠|—¯=ABCDE

chr$(27) corrisponde al comado:ESC

ESC+67 e' il comando della sottolineatura

ESC+68 e' il comando che la interrompe

Esempio si formula matematica

X con 7 elev.alla decima piu' X con 4 per Y elevato a (H piu' R)
meno X elevato alla J piu' Y:
$X_7^{10} + X_4 * Y^{(H+R)} - X^{(J+Y)}$

Ritorno alla stampa normale

Caratteri speciali per la stampa: chr$(X)

CHR$(30) per scrivere con caratteri PICA

(30+14) caratteri PICA allargati

CHR$(28) per scrivere in ELITE (17 c.p.p.)

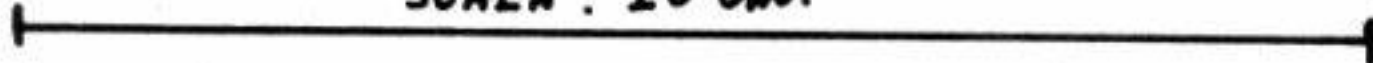
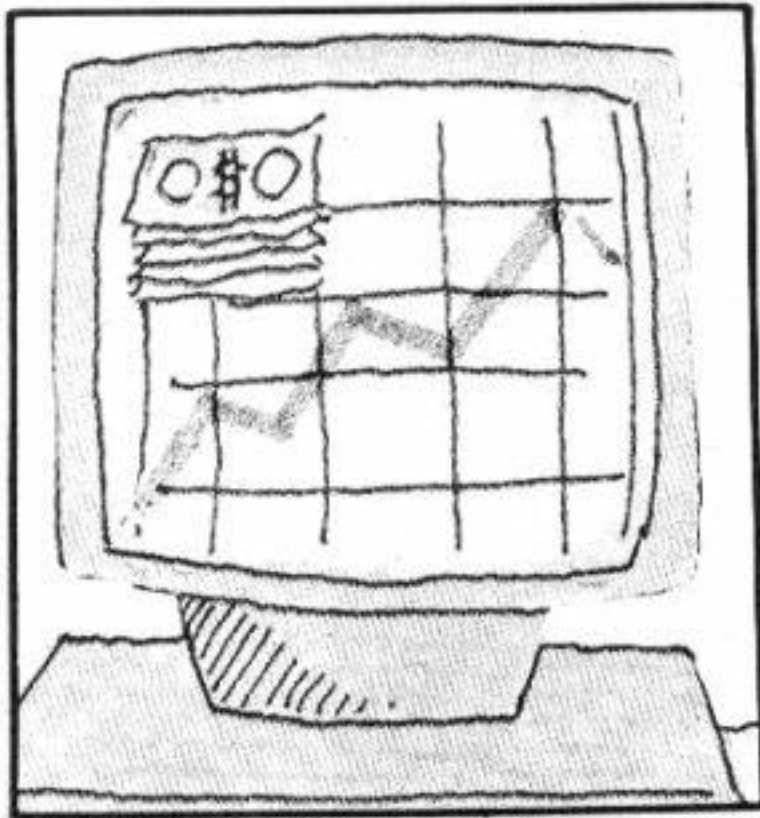
(28+14) caratteri ELITE allargati

CHR$(29) per scrivere CONDENSATO (12 c.p.p.)

(29+14) caratteri in CONDENSATO allargati

Con CHR$(15) si ritorna la modo di stampa PICA

Esempi di caratteri semigrafici COMMODORE:
ALFABETO:A♠ B C– D¯ E¯ F– G H I¬ J\ K⁄ L∟
M\ N/ O⌐ P¬ Q● R– S♥ T U/ V╳ WO X♣ Y Z♦

SCALA : 10 cm.

Due memorie molto, molto strane

Un'occasione unica per approfondire alcune nozioni riguardanti due puntatori fondamentali del Basic... e altre cose ancora.

di Claudio Baiocchi

Tra i messaggi d'errore che il Basic Commodore può inviare ai suoi utenti, uno dei più fastidiosi è ILLEGAL DIRECT.

Questo articolo spiega sia le sue motivazioni sia un trucco per evitarlo.

La trattazione è riferita a VIC-20 e C-64, ma sono descritte anche le (minime) varianti per C-16 e Plus/4. Perfino i possessori dei PET potranno seguire il discorso sostituendo ovunque le memorie 57 e 58 con le 54 e 55.

Che cosa sto facendo?

Nello svolgimento dei suoi compiti l'interprete Basic ha spesso la necessità di interrogarsi sull'operazione attualmente in corso. Ad esempio, quando incontra il codice 133, "token" (cioè simbolo) del comando INPUT, deve dapprima verificare se sta eseguendo una linea di programma (e in tal caso il comando sarà eseguito) o se il comando stesso è stato inserito in un comando diretto (e allora verrà emesso il messaggio ?ILLEGAL DIRECT).

Provate, infatti, a digitare direttamente:

```
INPUT A
```

e a premere il tasto Return.

Sorgono spontanee tre questioni.

- come fa il computer a sapere se sta elaborando una linea di programma oppure un comando diretto?
- Perchè i progettisti Commodore hanno previsto il divieto, in fase diretta, di eseguire alcuni comandi? Oltre a INPUT sono infatti rifiutati anche INPUT #, GET, DEF FN. Su C-16 e Plus/4 anche TRAP, RESUME, e GET # e GETKEY sono rifiutati come caso particolare di GET.
- E' possibile svincolarsi da tali restrizioni?

Una risposta affermativa alla terza domanda può essere particolarmente utile nel caso di errore in un programma che prevede l'accesso al disco. Quando la spia del drive comincia a lampeggiare, occorrerebbe interrogare il canale 15. Se, prevedendo guai, abbiamo inserito nel programma linee di help come le seguenti:

```
1000 CLOSE 15:OPEN15,8,15
1010 INPUT # 15,A,B$,C,D
1020 CLOSE15:PRINT A,B$,C,D
```

basterà fare: GOTO 1000.

I possessori di C-16 e Plus/4 ottengono le stesseiinformazioni digitando semplicemente PRINT DS$.

Tuttavia spesso si ha troppa fiducia nei propri programmi, non si prevedono errori e non si predispongono linee di help, ed è chiaro che, se si digita il minilistato proposto a guaio avvenuto, si perdono tutti i dati elaborati fino a quel momento. Mentre, provando a eseguire in forma diretta i vari comandi, si otterrà solo un ?ILLEGAL DIRECT...

Le locazioni 57 e 58

Per dare una (parziale) risposta alla prima domanda basta leggere in ROM le routine LM (linguaggio macchina) che gestiscono i comandi proibiti.

Sul Vic 20 lutte iniziano con un JSR $D3A6; sul C-64 con JSR $B3A6. Per C-16 e Plus/4 si veda più avanti.

Chi è digiuno di linguaggio Assembler, dovrà solo tener presente che l'istruzione JSR è l'equivalente del GOSUB in Basic. All'indirizzo in questione troviamo poi le istruzioni descritte, tradotte e commentate in riquadro 1.

Assembler	Semi-traduzione Basic
$D3A6 LDX $3A	X=PEEK(58)
$D3A8 INX	X=X+1
$D3A9 BNE $D3B4	IF X=256THEN RETURN
$D3AB LDX# $15	X=21
$D3AD BIT $1BA2	(Vedi nota)
$D3B0 JMP $c437	GOTO ROUTINE ERRORE
$D3B4 RTS	RETURN

La routine di errore stamperà il messaggio N.X che, ad esempio, per X=21, è ?ILLEGAL DIRECT.

Nota: la $D3AD è un trucco per risparmiare spazio: altrove in ROM con JMP $D3AE (oppure BNE $d3AE e simili) si otterrà l'effetto di:

```
LDX # $1B
JMP $C437
```

Riquadro 1 La routine ROM che, sul Vic 20, vieta di usare, in forma diretta, alcuni comandi. Sul C-64 tutti gli indirizzi vanno diminuiti di $2000. Su C-16 e Plus/4 la routine è un po' diversa e usa la memoria $81 anzichè $3A.

Pur senza conoscere il meccanismo col quale l'interprete Basic gestisce la locazione 58, possiamo comunque formulare una regola importante:

Regola n. 1

La fase di esecuzione di comandi diretti è caratterizzata dalla validità della relazione:

PEEK(58)=255

Su C-16 e Plus/4, oltre alla relazione vista, si ha anche PEEK(129)=0 ed è su quest'ultima locazione che viene effettuato il controllo.

Sempre il contenuto della locazione 129 è responsabile del messaggio ?DIRECT MODE ONLY emesso quando, nell'elaborazione di un programma, è incontrato un RENUMBER, un DELETE oppure un AUTO.

Proviamo ora a fare i primi tentativi di "truffa" all'interprete riferendoci d'ora in poi a Vic 20 e C-64. Le varianti dovrebbero essere ovvie: digitiamo, tutto di seguito, su uno stesso rigo:

```
A$="":POKE57,0:POKE58,4:
INPUTA$:PRINTA$:STOP
```

Chi ha il C-16 oppure il Plus/4 dovrà aggiungere la Poke relativa alla locazione 129, cioè:

```
A$="":POKE57,0:POKE58,4:POKE
129,128:INPUTA$:PRINTA$:STOP
```

Premendo il tasto Return, a conferma dell'avvenuto inganno, vedremo apparire il punto di domanda (?), rappresentante la richiesta di Input.

I guai che possono avvenire fornendo un A$ troppo lungo sono descritti nel riquadro 2. Per ora limitiamoci a constatareiche, fornendo in risposta un A$ "corto", tutto funziona regolarmente(?): il dato viene accettato (Input), stampato (Print) e seguito da un BREAK IN 1024.

Si digiti, tutto di seguito e senza spazi intermedi:
A$="":POKE57,0:POKE58,4:INPUTA$:PRINTA$:STOP
e si prema Return. Le POKE iniziali fanno ritenere all'interprete di stare elaborando la linea Basic 1024, e non un comando diretto. Pertanto la richiesta di input è eseguita.

FORNENDO COME A$:	SI OTTIENE:
una stringa con LEN(A$)<19	A$ è accettato e stampato segue BREAK IN 1024
la stringa di 19 caratteri ABCDEFGHIJKLMNOPQRS	A$ non è accettato; segue ?SYNTAX ERROR IN 1024
la stringa di 20 caratteri ABCDEFGHIJKLMNOPQRST	A$ non è accettato; segue ?REDO FROM START, poi ?SYNTAX ERROR IN 1024
la stringa di 21 caratteri ABCDEFGHIJKLMNOPQRST$	resta A$="". La stringa digitata è immessa, stranamente, in T$ segue BREAK IN 14884

Sul Vic 20 e C-64. Per le varianti relative a C-16 e Plus/4, si veda l'articolo.

Riquadro 2 - L'interprete, imbrogliato, reagisce in modo strano.

Proviamo poi a modificare la linea precedente scegliendo altri valori da pokare in 57 e 58: sempre con A$ corti, ci accorgiamo che il numero che segue BREAK IN ha sempre la forma:
PEEK(57)+256*PEEK(58)
A complemento della regola 1 possiamo perciò formulare la regola n. 2.

Regola n. 2
Se è PEEK(58)<255, l'interprete Basic ritiene di stare elaborando la linea Basic numero:
PEEK(57)+256*PEEK(58)

Si osservi che le due regole non sono in contrasto tra loro: salvo operazioni... piratesche dell'utente, il numero di una linea Basic vale al massimo 63999 (=255+256*249). In fase di elaborazione di un programma sarà perciò sempre PEEK(58)<250.

Un esame più attento

Gli "strani" comportamenti visti nel riquadro 2 indicano che, per dare una risposta efficiente al terzo quesito, occorre preliminarmente rispondere alla seconda domanda. La cosa non è difficile, ma richiede alcune premesse su cosa avviece quando digitiamo qualcosa e premiamo il tasto Return. Precisamente l'interprete esegue leitre operazioni:

- ricopia la frase nel buffer Basic (locazioni di memoria da 512 a 600);
- terminata l'operazione aggiunge uno zero (0) a segnalare che il messaggio è finito (non c'è pericolo di confusione con la cifra "0": questa viene trascritta tramite 48, suo codice ASCII);
- soppressi eventuali spazi, o caratteri grafici iniziali, il contenuto del buffer viene "compresso", sostituendo le parole chiave con i relativi token.

L'operazione viene svolta "sul posto", cioè il nuovo messaggio si sovrappone via via al messaggio già presente nel buffer che rappresenta, carattere per carattere, ciò che abbiamo realmente digitato.

Non ci sono pericoli di sovrapposizioni perchè in qualsiasi caso il nuovo messaggio è lungo al massimo quanto il precedente. Nel caso di comandi Basic, infatti, questi vengono tokenizzati e occupano quindi un minor numero di byte. Nel caso di caratteri alfanumerici privi di significato Basic, occuperanno lo stesso posto di prima.

Quando si trova lo zero lo si trascrive; poi, saltata una casella, si scrive un altro 0 il cui ruolo è analogo a quello della terna di zeri che chiude ogni programma Basic.

Se PEEK(512), cioè la prima locazione del buffer, è compreso tra 48 e 57, vuol dire che il messaggio digitato inizia con una cifra. Si deduce che l'utente ha digitato una linea Basic, da modificare, sopprimere o creare ex-novo inserendola al posto giusto. In caso contrario, si tratta di comandi diretti, da eseguire immediatamente.

Per rispondere alla seconda domanda, dunque, basta tener presente che, in occasione di INPUT, GET e simili, i dati in input transitano preliminarmente per il buffer Basic (a partire dalla cella 512, e terminanti con uno 0 finale) per la successiva elaborazione.

E' allora chiaro che, se INPUT e simili venissero accettati in un comando diretto, i dati in input potrebbero "sporcare" il comando che li ha richiesti, sovrapponendosi ad esso nel buffer!

Per l'esecuzione di DEF, invece, è necessario tener presente che si prevede la memorizzazione, nella zona variabili, del punto di inizio della formula che definisce la funzione. E' fin troppo ovvio che se tale formula è scritta solo nel buffer Basic, essa sarà resa inutilizzabile dai successivi comandi diretti.

Conclusioni

Volendo usare in forma diretta uno dei comandi proibiti occorrerà, oltre alla modifica della locazione 58, o della 129 su C-16 e Plus/4, porre tale comando alla fine del buffer.

Ad esempio, per colloquiare in forma diretta col canale 15 del disco, dopo un cautelativo:
CLOSE15:OPEN15,8,15
basterà digitare, tutto di fila, ricorrendo alle abbreviazioni note (DATA=Da; PRINT=?, eccetera):
DATA"QUESTE POSIZIONI POSSONO ESSERE SPORCATE": POKE58,0: INPUT # 15,A,B$,C,D: PRINTA,B$,C,D
e premere Return.

Si osservi però che la risposta data alla prima domanda è ancora incompleta: le regole 1 e 2 ci dicono come il Commodore utilizza i contenuti delle locazioni 57 e 58, ma non come fa a predisporre tali contenuti.

Avendo in memoria il programma A, eseguendo RUN 30 oppure RUN 40 si otterrà ?SYNTAX ERROR IN 40. Eseguendo invece RUN, RUN 10 oppure RUN 20 si otterrà ?UNDEF'D STATEMENT, seguito da ERROR IN 10. Eseguendo infine RUN 50 (oppure RUN 60, eccetera) si otterrà ?UNDEF'D STATEMENT seguito solo da ERROR (senza numero di linea dopo ERROR).

Programma A

```
10 GOTO 60
20 GOTO 10
30 REM
40 PEEK
50 POKE 58,255:
GOTO 60
```

Programma B

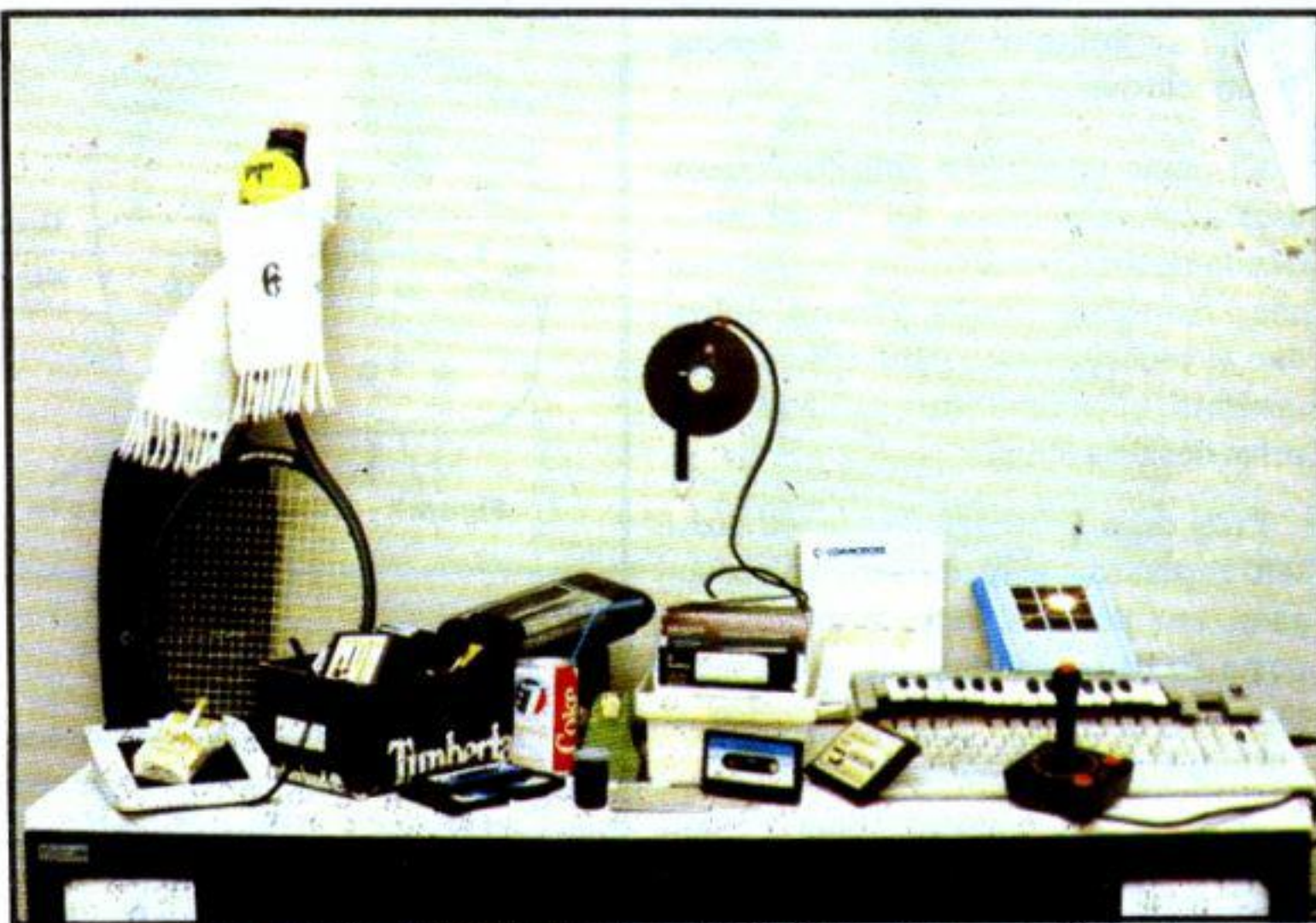
```
10 REM
20 POKE 57,10: GOTO 20
30 POKE 57,40 : PEEK
40 GOTO 30
```

Per quanto strano possa sembrare (in particolare nel caso di RUN 20) una diagnostica identica si ottiene avendo in memoria il programma B.

Riquadro 3 - Manipolando le locazioni 57 e 58 la diagnostica risulta stravolta.

80 colonne sull'apparecchio TV

di **Giancarlo Mariani**

Per sfruttare bene il Commodore 128 è sicuramente utile il monitor speciale RGB da 80 colonne che ha però un difetto notevole: occorre pagarlo...

Sì, avete capito bene! Proprio 80 (ottanta) colonne su un normalissimo televisore domestico oppure, se lo possedete, sul "solito" monitor monocromatico.

A questo punto i lettori cominceranno ad arricciare il naso: "si tratterà della solita interfaccia multiuso e multi...prezzo o, peggio, di un programma dalle dubbie qualità in fatto di definizione dell'immagine (vedi prodotti analoghi per Vic-20 e C-64)".

Niente di tutto questo! Per visualizzare le 80 colonne non è necessaria nessuna particolare (!) interfaccia, nè tantomeno un programma: bastano soltanto due fili elettrici e un qualsiasi modulatore Tv.

Costo dell'operazione: praticamente nullo, calcolando che il modulatore si può recuperare da una vecchia console per video-giochi in disuso, oppure da un amico hobbysta elettronico. Al limite, si può utilizzare quello del Vic-20 nel caso abbiate ancora il simpatico computer della Commodore che, dopo l'acquisto del C-128, avete forse dimenticato in cantina.

A chi possiede un monitor non sarà neanche necessario il modulatore: ci si può limitare ad acquistare soltanto un cavetto schermato e una presa Cannon eguale, come piedinatura, a quella montata sul joystick.

Com'è possibile visualizzare 80 colonne su una Tv se l'uscita del C-128 è addirittura RGB? C'è forse qualche inghippo?

Ovviamente bisogna tener conto che la qualità dell'immagine non è delle migliori e che, tra l'altro, non è possibile "uscire" a colori. Per il resto, però, non ritiriamo una parola di quanto affermato. Precisiamo comunque che questa non è una soluzione *alternativa* al monitor RGB, ma solo una soluzione *provvisoria*, che permette tuttavia di usare decentemente programmi sviluppati per le 80 colonne, in attesa dell'acquisto di un monitor adeguato.

Come realizzare il miracolo

Veniamo alla descrizione delle operazioni da compiere.

La "scoperta" che si sta per descrivere è avvenuta quasi per caso. Appena venuti in possesso del tanto desiderato C-128, ci è subito venuta voglia di provare il CP/M a 80 colonne e, non potendo provvedere anche all'acquisto di un monitor RGB ci siamo messi al lavoro per cercare una soluzione alternativa.

L'idea è nata osservando attentamente la piedinatura della porta RGB (a pagina C-6 del manuale fornito col computer). E' presente, infatti, uno strano contatto, indicato col termine "monochrome".

Ci siamo affrettati a collegare questa uscita col modulatore del Vic-20 attendendo risultati positivi sulla Tv. Ed ecco che... sorpresa! Non funzionava. Sul video si vedevano solo righe oblique che, in effetti, non erano altro che l'immagine priva dei sincronismi.

Presi dalla disperazione e ormai quasi senza speranza, abbiamo provato a collegare, una per una, tutte le uscite della porta RGB col modulatore: a un certo punto è comparso sull'apparecchio il "prompt" delle 80 colonne del C-128.

Il piedino incriminato è "horyzontal sync", sincronismo orizzontale e non chiedetemi perchè funziona, ma funziona.

Basta infatti prelevare il segnale tra quel contatto e la massa e poi "sbatterlo" in un modulatore e quindi alla Tv.

Pèr i possessori di un monitor, come detto prima, non sarà necessario il modulatore: basterà infatti collegare direttamente l'uscita "horizontal sync" con l'entrata del monitor.

Per chi non vuole fare la "fatica" di cercare o acquistare un modulatore, verrà presentato su uno dei prossimi numeri uno schema elettrico di semplice realizzazione, adatto per qualsiasi tipo di televisore domestico e... per qualsiasi lettore, anche se alle prime armi col saldatore.

Le connessioni da effattuare sono ben visibili nelle figure di queste pagine.

Un'avvertenza: non mandare alla Tv i due segnali (40 e 80 colonne) contemporaneamente, perchè si disturberebbero l'un l'altro. L'ideale è usare un deviatore, con il quale scegliere, a seconda delle esigenze, l'uno o l'altro sagnale.

Per evitare interferenze si raccomanda l'uso di cavetti schermati del tipo televisivo. La qualità dell'immagine (come detto prima monocromatica, ossia in bianco e nero anche su Tv a colori) non è proprio eccezionale e dipende comunque dal tipo di televisore ulilizzato e dalla combinaziona di colori sfondo/bordo selezionata.

Gli effetti migliori si ottengono con televisori o monitor b/n di media grandezza (12-16 pollici) e con caratteri scuri su sfondo chiaro. I risultati ovviamente non sono assolutamente paragonabili a quelli forniti da un monitor RGB ma, come dice il proverbio, chi si accontenta...

Pin	Signal
1	Gnd
2	Gnd
3	Red
4	Green
5	Blue
6	Intensity
7	Monocprome
8	Hor. Sync
9	Vert. Sync

Figura 1 - Connessioni della porta RGB C-128.

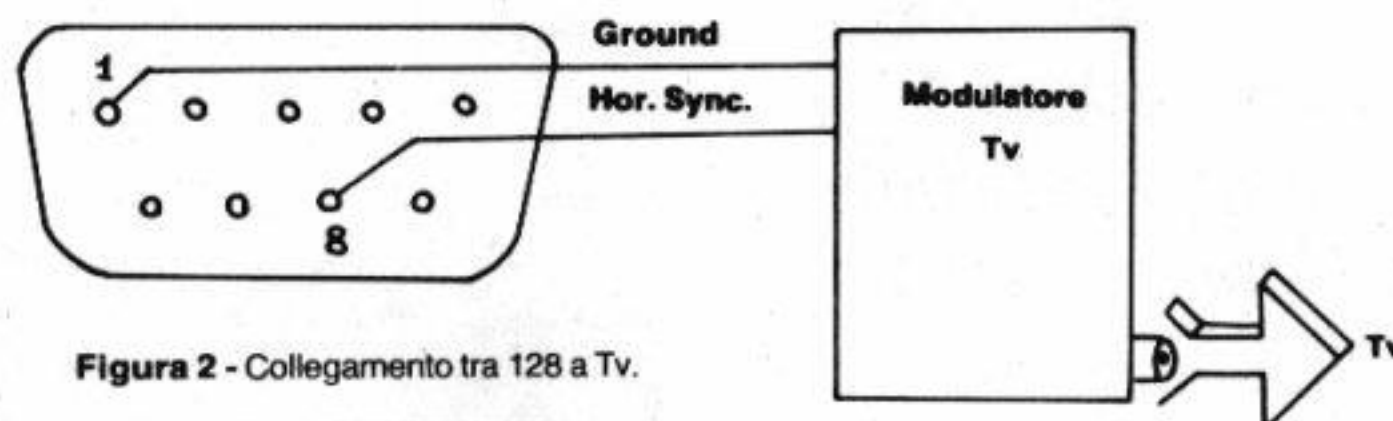



Figura 2 - Collegamento tra 128 a Tv.

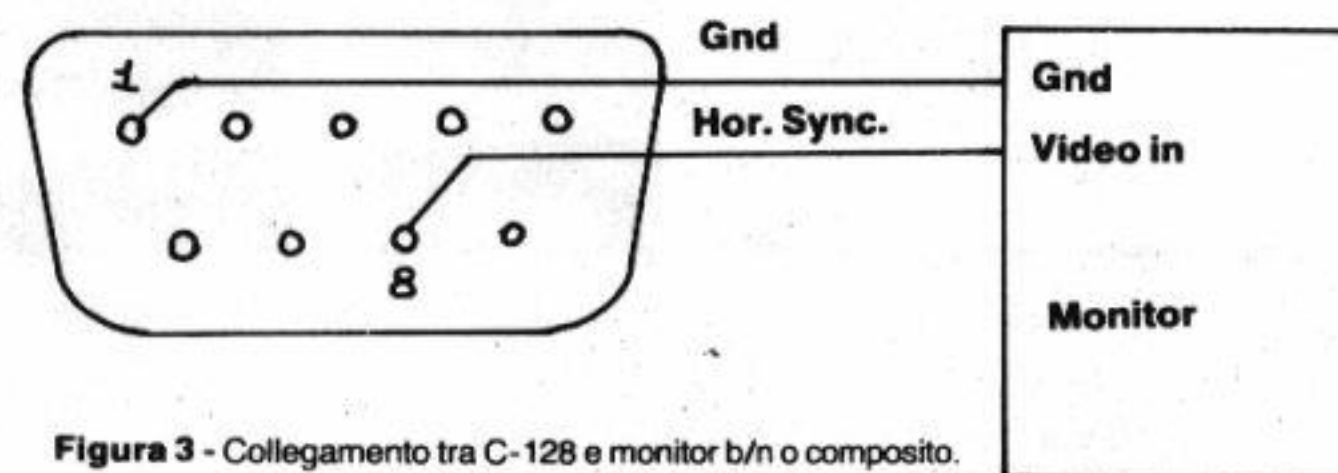



Figura 3 - Collegamento tra C-128 e monitor b/n o composito.

Chi ha ucciso Michael Jackson?

Indossate i panni di Sherlock Holmes per risolvere un enigma degno del miglior autore di libri gialli...

di Antonio Visconti

Un colpo d'arma da fuoco riecheggia nell'aria. Nel corridoio, su cui affacciano i camerini degli artisti, giace il corpo senza vita di Michael Jackson.

Qualcuno corre, ma non è possibile vedere chi è Immediatamente tutte le uscite vengono bloccate, l'assassino non può fuggire.

Solo una tra sei persone può aver compiuto l'infame crimine. Con un pressante interrogatorio bisogna scoprirla e in fretta: la stampa e l'opinione pubblica vogliono il colpevole.

L'interrogatorio

Ci deve essere un motivo per compiere un omicidio. Una delle chiavi per risolvere il caso è la ricerca di chi ha un movente. Non basta, ci vuole una pistola e bisogna, ovviamente, trovarsi sul luogo del delitto.

"Dove si trovava al momento dello sparo?"

Ecco una delle domande che anche il più sprovveduto investigatore pone al sospetto che gli si trova davanti.

Prendiamo nota delle risposte, abbiamo trovato uno che odiava il morto (movente), possiede una pistola (arma), ma era lontano da dove sono successi i fatti.

Dobbiamo ricominciare daccapo. Passiamo una mano sulla nostra barba ispida (ci hanno svegliato alle quattro del mattino, orario ideale per trasformare in carogna uno che se ne stava per i fatti suoi) e ricominciamo.

Il colpevole è colui (colei) che ha movente, arma e opportunità e solo in presenza di tutti e tre i requisiti possiamo formulare l'accusa con sicurezza e schiaffare l'assassino dietro le sbarre.

Continuiamo a fare domande: due degli interrogati hanno dato risposte contraddittorie o forse siamo noi che all'alba non siamo ancora completamete svegli. Uno dei due è un bugiardo e alla fine glielo diremo in faccia, quella lurida faccia da topino sott'olio in vacanza.

(N.B. Il linguaggio deve essere adeguato al clima.)

Dobbiamo interrogare un terzo testimone e se anche questo mente? Vorremmo mandare tutto al diavolo ed essere tra le lenzuola, ma la "lei" russa ed è meglio esser qui, in compagnia di un assassino sconosciuto, mondo boia.

Il caso è ingarbugliato, ci vuole arguzia, logica, spirito di osservazione e un'assicurazione sulla vita: quel figlio di koala (o si dice di cane?) può colpire ancora.

Sappiamo soltanto che uno dei sospetti mente sempre, un altro solo qualche volta. Il bugiardo non è necessariamente il colpevole. E' un caso interessante, maledettamente interessante...

Il programma

Il colpevole, il bugiardo e il mezzo bugiardo (che possono coincidere o meno) vengono scelti casualmente all'inizio del gioco. Quindi l'indagine si ripete sempre in maniera diversa ogni volta che si digita RUN. All'inizio compare un menu che vi permette di scegliere tra l'interrogare i sospetti, riepilogare le risposte ottenute sino a quel momento e formulare un'ac-

cusa, da utilizzare quando si è finalmente scoperto l'assassino.

Si possono fare tre tipi di domande, al fine di accertare il movente, il possesso di un arma e l'opportunità di compiere il misfatto. Il programma vi chiede chi volete interrogare, poi su quale dei sospetti volete informazioni (non è però possibile chiedere a uno di loro qualcosa su se stesso).

La logica di funzionamento del programma è molto semplice. Viene preparata una tabella di sei righe e tre colonne, come quella di figura 1, in cui su ogni riga vi è uno dei sospetti e sulle colonne i tre parametri chiave: il movente, l'arma e l'opportunità.

P$(6) contiene i nomi dei personaggi.

MV$(4) contiene le risposte sul movente.

OP$(6) le risposte sull'opportunità.

R$(20) le risposte degli interrogati (riepilogo).

Questa fase termina alla linea 105.

Alle linee 110-125 vengono scelti casualmente il colpevole, il bugiardo e il mezzo bugiardo e viene di conseguenza preparata la tabella di verità.

Viene allora stampato il menu (linee 135-150) e, a seconda della scelta, si eseguono i sottoprogrammi relativi.

L'interrogatorio, le cui modalità di svolgimento abbiamo già spiegato, viene realizzato alle linee 165-240.

	personaggio	movente	arma	opport.
1	Madonna	1	1	1
1	M. Jagger	0	1	0
0	D. Bowie	1	1	0
1	T. Turner	0	0	1
2	Bob Dylan	0	0	1
1	Sting	1	0	1

Figura 1 - La tabella di verità. Il colpevole è Madonna, il bugiardo D. Bowie, il mezzo bugiardo Bob Dylan.

Il colpevole è l'unico con tutti e tre i parametri posti ad 1 (Madonna, nell'esempio in figura). Quando formuliamo una domanda, viene fatto un salto nella prima colonna della tabella.

- Nel caso vi si trovi un uno (1) (Madonna, Jagger, Turner, Sting) viene scelta, casualmente, una delle possibili risposte di conferma.
- Nel caso di uno zero (Bowie), una delle risposte di smentita. Se l'interrogato è un bugiardo la situazione precedente si inverte, conferma nel caso di 0, smentita nel caso di 1. Per il mezzo bugiardo (Dylan) la scelta tra smentita e conferma è completamente casuale.

Uno sguardo al listato

Il diagramma di flusso è riportato in figura 2. La fase di inizializzazione consiste nello scegliere i colori di sfondo (blu) e di dimensionamento degli arrays.

TV(6,3) è la Tabella di verità.

Il riepilogo alle linee 425-450.

L'accusa alle linee 315-420.

Se il caso si conclude felicemente, viene assegnato un punteggio che tiene conto del numero di domande fatte e del tempo impiegato. In caso di errore il riepilogo vi permetterà di capire perchè avete sbagliato.

Le linee 460-645 non necessitano di spiegazione; le ultime quattro infine realizzano un effetto sonoro.

Possibili sviluppi

Il programma offre molteplici spunti per un ulteriore sviluppo. Si possono aumentare il numero di possibili risposte, cambiare scenario, cambiare i personaggi. Questo tipo di modifiche non aumenta la difficoltà del gioco, che può essere incrementata, ad esempio, aumentando il numero di sospetti. Attenzione, però, che se si aumenta solo il numero di quelli che dicono la verità, diminuisce il peso dei bugiardi; quindi è opportuno aumentare anche il numero di questi ultimi. Il programma riconosce per bugiardo il personaggio che ha uno 0 nella colonna 0 della tabella di verità.

Se TV(I,0)=0 allora l'Iesimo personaggiot bugiardo, per il mezzo bugiardo TV(I,0)=2.

Figura 2 - Diagramma di flusso.

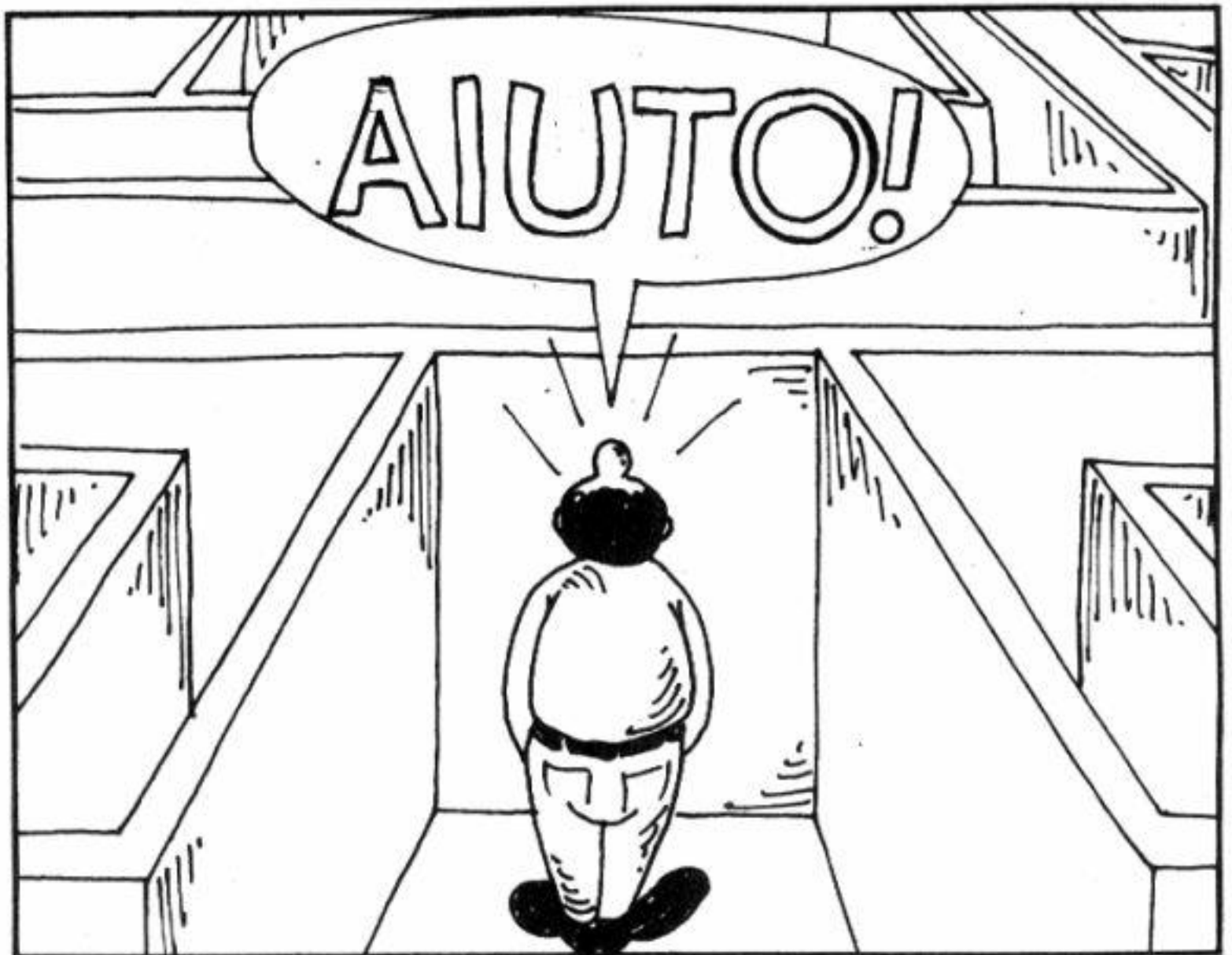
```
10 REM  CHI HA UCCISO
15 REM  MICHAEL JACKSON?
20 :
25 REM  DI ANTONIO VISCONTI
30 REM  PER QUALSIASI COMMODOR
   E
35 :
40 REM  I POSSESSORI DI COMPUT
   ER DIVERSI
45 REM  DAL C-64 DEVONO CAMBIA
   RE
50 REM  IL NUMERO CHE SEGUE CO
   N UNO
55 REM  QUALSIASI
60 :
65 COMPUTER=64
70 :
75 REM  **********************
   ***********
80 IF CO=64 THEN REM  INSERIRE
    LE POKE DI SCHERMO PER IL
   C-64
85 :
90 REM  INSERIRE QUI I COMANDI
    DI COLORE BORDO E SCHERMO
   PER C-16 E VIC 20
95 :
100 DIM TV(6,3),P$(6),MV$(4),OP
   $(6),R$(23):FL=RND(-TI/49):
   GOSUB 460
105 GOSUB 545:FR=200:FOR I=1 TO
    6:TV(I,0)=1:NEXTI
110 KL=INT(RND(0)*6+1):FOR I=1
   TO 3:TV(KL,I)=1:NEXTI
115 FL=INT(RND(0)*6+1):TV(FL,0)
   =0:FL=INT(RND(0)*6+1):TV(FL
   ,0)=2
120 FOR I=1 TO 6:J=INT(RND(0)*3
   +1):TV(I,J)=1
125 J=INT(RND(0)*3+1):TV(I,J)=1
   :NEXTI
130 TI$="000000":TM=TI
131 UR=0:FOR I=1 TO 6:IF TV(I,0
   )=2 THEN UR=1
132 NEXT:FOR I=1 TO 6:IF TV(I,0
   )=0 THEN UR=UR+1
133 NEXT:IF UR<>2 THEN  RUN
135 PRINTCHR$(147)"CHE COSA VUO
   I FARE ?"
140 PRINTCHR$(17)"[RVS] 1[RVOFF
   ] INTERROGARE I SOSPETTATI"
145 PRINT"[RVS] 2[RVOFF] RIEPIL
   OGARE"
150 PRINT"[RVS] 3[RVOFF] FORMUL
   ARE UN'ACCUSA"
155 GET A$:ON VAL(A$)GOTO 165,4
   25,315
160 GOTO 155
165 GOSUB 650:IF K>20 THEN 245
170 PRINTCHR$(147)"CHI VUOI INT
   ERROGARE ?":PRINT:PRINT:PRI
   NT
175 FOR I=1 TO 6:PRINT:PRINT"[R
   VS]";I;"[RVOFF]";" ";P$(I):
   NEXTI
180 GET A$:TS=VAL(A$):IF TS<1 O
   R TS>6 THEN 180
185 GOSUB 650:PRINTCHR$(19)"SU
   QUALE DEI SOSPETTATI":PRINT
   "VUOI INFORMAZIONI ?"
190 GET A$:SP=VAL(A$):IF SP<1 O
   R SP>6 OR SP=TS THEN 190
195 GOSUB 650:PRINTCHR$(147)"QU
   ALE E' LA TUA DOMANDA ?"
200 PRINTCHR$(17)"[RVS] 1[RVOFF
   ] ";P$(SP);" AVEVA UN MOVEN
   TE ?"
205 PRINT"[RVS] 2[RVOFF] ";P$(S
   P);" POSSIEDE UN'ARMA ?"
210 PRINT"[RVS] 3[RVOFF] DOVE "
   ;P$(SP);" SI TROVAVA ?"
215 GET A$:R=VAL(A$):IF R<1 OR
   R>3 THEN 215
220 GOSUB 650:ON RGOSUB 255,275
   ,295:PRINT:PRINT
225 R$(K)=P$(TS)+" DICE CHE "+P
   $(SP):PRINTR$(K):PRINTTB$:R
   $(K)=R$(K)+TB$
230 K=K+1:PRINTCHR$(17)"PREMI U
   N TASTO"
235 GET A$:IF A$="" THEN 235
240 GOTO 135
245 PRINTCHR$(147)"TROPPE DOMAN
   DE:":PRINT"AVRESTI GIA' DOV
   UTO INDOVINARE!"
250 CL=100:GOTO 340
255 TB$="":IF TV(SP,1)=TV(TS,0)
    THEN 270
```

```
260 IF TV(TS,0)=2 THEN FL=RND(0
    ):IF FL<.5 THEN 270
265 TB$=" NON"
270 FL=INT(RND(0)*4+1):TB$=TB$+
    MV$(FL):RETURN
275 TB$="":IF TV(SP,2)=TV(TS,0)
     THEN 290
280 IF TV(TS,0)=2 THEN FL=RND(0
    ):IF FL<.5 THEN 290
285 TB$=" NON"
290 FL=INT(RND(0)*4+1):TB$=TB$+
    " POSSIEDE UNA PISTOLA":RET
    URN
295 IF TV(SP,3)=TV(TS,0) THEN 3
    10
300 IF TV(TS,0)=2 THEN FL=RND(0
    ):IF FL<.5 THEN 310
305 FL=INT(RND(0)*3+1):TB$=OP$(
    FL):RETURN
310 FL=INT(RND(0)*3+4):TB$=OP$(
    FL):RETURN
315 GOSUB 650:PRINTCHR$(147)"CH
    I E' SECONDO TE L'ASSASSINO
     ?"
320 PRINT:PRINT:PRINT:FOR I=1 T
    O 6:PRINT:PRINT"[RVS]";I;"[
    RVOFF]";" ";P$(I):NEXTI
325 GET A$:CL=VAL(A$):IF CL<1 O
    R CL>6 THEN 325
330 IF CL=KL THEN 390
335 PRINTCHR$(147)"    SBAGLIATO
     !"
340 PRINT" IL COLPEVOLE E' ";P$
    (KL)
345 PRINT"PERCHE' HA MOVENTE,AR
    MA ED OPPORTUNITA'"
350 FOR I=1 TO 6:IF TV(I,0)=0 T
    HEN BG=I
355 IF TV(I,0)=2 THEN MB=I
360 NEXTI
365 PRINT" IL BUGIARDO E' ";P$(
    BG)
370 PRINT" IL MEZZO BUGIARDO ";
    P$(MB)
375 PRINTCHR$(17)"PREMI UN TAST
    O"
380 GET A$:IF A$="" THEN 380
385 LA=1:GOTO 425
390 PRINTCHR$(147)"[2 DOWN] ESA
    TTO !"
395 PRINT"COMPLIMENTI SEI UN OT
    TIMO INVESTIGATORE"
400 PRINT"HAI INDOVINATO DOPO "
    ;K;" DOMANDE"
405 PRINT"IL TEMPO IMPIEGATO E'
     STATO DI "
410 PRINT"ORE ";LEFT$(TI$,2);"
    MIN. ";MID$(TI$,3,2);" SEC.
     ";RIGHT$(TI$,2)
415 PRINT"CONQUISTI PUNTI ";200
    0-K*100-(TI-TM)/60
420 END
425 GOSUB 650:PRINTCHR$(147):FO
    R I=0 TO 10:PRINTR$(I):NEXT
    I:PRINT"PREMI UN TASTO"
430 GET A$:IF A$="" THEN 430
435 IF K<11 THEN 450
440 PRINTCHR$(147):FOR I=11 TO
    20:PRINTR$(I):NEXTI:PRINT"P
    REMI UN TASTO"
445 GET A$:IF A$="" THEN 445
450 IF LA THEN END
455 GOTO 135
460 P$(1)="MADONNA"
465 P$(2)="MICK JAGGER"
470 P$(3)="DAVID BOWIE"
475 P$(4)="TINA TURNER"
480 P$(5)="BOB DYLAN"
485 P$(6)="STING"
490 MV$(1)=" ODIAVA M.J."
495 MV$(2)=" AVEVA UN BUON MOTI
    VO PER UCCIDERE"
500 MV$(3)=" PROVAVA INVIDIA PE
    R M.J."
505 MV$(4)=" DOVEVA DEI SOLDI A
     M.J."
510 OP$(1)=" ERA NELLA PROPRIA
    STANZA "
515 OP$(2)=" ERA ANDATO A DORMI
    RE PRESTO "
520 OP$(3)=" FORSE ERA IN CUCIN
    A A FARE UNO SPUNTINO "
525 OP$(4)=" NON ERA NELLA PROP
    RIA STANZA "
530 OP$(5)=" ERA COL MORTO FINO
     A POCO PRIMA DELLO"+"   "+"
    SPARO"
535 OP$(6)=" ERA... CHISSA' DOV
```

```
    E "
540 RETURN
545 PRINTCHR$(147)"UN COLPO D'A
    RMA DA FUOCO"
550 PRINT"RIECHEGGIA NELL'ARIA.
    "
555 PRINT"NEL CORRIDOIO GIACE U
    N CORPO SENZA VITA."
560 PRINT"UN RUMORE DI PASSI"
565 PRINT"CI DICE CHE QUALCUNO
    STA FUGGENDO."
570 PRINTCHR$(17)"PREMI UN TAST
    O"
575 GET A$:IF A$="" THEN 575
580 PRINTCHR$(147)" CHI HA UCCI
    SO"
585 PRINT"MICHAEL JACKSON ?"
590 PRINTCHR$(17)"PREMI UN TAST
    O"
595 GET A$:IF A$="" THEN 595
600 PRINTCHR$(147)" I SOSPETTI
    SONO CONCENTRATI SU"
605 PRINT"SOLO 6 PERSONE"
610 PRINT"A TE IL COMPITO DI SC
    OPRIRE L'ASSASSINO"
615 PRINT"ATTENZIONE !":PRINT
620 PRINT"- UNO DEI SOSPETTI ME
    NTE SEMPRE"
625 PRINT"- MENTRE UN ALTRO SOL
    O QUALCHE VOLTA"
630 PRINT:PRINT"NON FARTI INGAN
    NARE !"
635 PRINTCHR$(17)"PREMI UN TAST
    O"
640 GET A$:IF A$="" THEN 640
645 RETURN
650 IF  CO<>64  THEN  RETURN
655 FOR I=54272 TO 54295:POKE I
    ,0:NEXT:POKE 54296,15
660 W=54276:POKE  W+2,9:POKE  W
    -3,FR:POKE  W,23:POKE  W,22
665 FOR  I=0  TO  100:NEXT:RETU
    RN
```

Costruisci un labirinto!

Un videogioco divertente, semplice da comprendere e molto breve. Utile, soprattutto, a chi vuole capire come manovrare gli sprites e generare un suono.

di Maurizio Dell'Abate

Questi programmi possono girare (funzionare) soltanto sul Commodore 64. Non è possibile, in BASIC, un adattamento ad altri Commodore, essendo il gioco imperniato su una caratteristica esclusiva del Commodore 64: gli sprites. I possessori di altri computer possono tuttavia leggere quelle parti dell'articolo valide per tutti i Commodore, come il funzionamento di alcune istruzioni BASIC o particolari tecniche di progammazione.

Come inventare un gioco

Quest'articolo è rivolto a quanti vorrebbero programmare un videogioco, ma non posseggono sufficiente esperienza per farlo. Si è evitato quindi di approfondire ulteriormente argomenti che risulterebbero troppo complessi e che darebbero luogo, al contrario, a un'improduttiva confusione.

L'idea di base

Con una sola riga (vedi primo listato) è possibile tracciare sul video labirinti casuali che sembrano accuratamente progettati e piuttosto complessi.
La routine visualizza in successione una delle due sbarrette (caratteri semigrafici) che si ottengono premendo SHIFT + N oppure SHIFT + M. La sbarretta da visualizzare viene scelta casualmente e si ottengono labirinti simili a quello di figura 5.

Le videate che ne risultano sembrano suggerire un gioco in cui è necessario percorrere disordinati corridoi e, ovviamente, raggiungere l'uscita.

Si è pensato quindi di realizzare un videogame il cui scopo principale sia quello di risolvere (usiamo pure questo termine) un labirinto. Scartata a priori la possibilità di percorrerlo facendo scorrere un dito sul video (ci sarebbe tra l'altro la necessità di trapassare i muri!), si è scelta la soluzione più razionale: utilizzare un minuscolo sprite che si muove sotto il comando di quattro tasti.

Sprite: chi era costui?

Per coloro che non sanno cosa sia uno sprite, chiariamo subito: gli sprites sono figure disegnate dall'utente posizionabili in qualsiasi punto del video. Si possono sovrapporre ai caratteri senza cancellarli e sono del tutto indipendenti dal "testo". E' possibile leggere anche una collisione con il testo sottostante (il nostro omino con i muri del labirinto) e far agire il programma di conseguenza.

Si possono visualizzare fino ad otto sprites contemporaneamente (numerati da 0 a 7), ma tutto ciò che sarà detto in questo articolo si riferisce allo sprite numero zero. Si consiglia tuttavia a coloro che di sprites sanno poco o niente, di leggere attentamente le (poche) nozioni

riportate sul manuale del C-64 per seguire meglio il resto di quest'articolo.

Un'immagine più precisa del gioco

Cerchiamo ora di crearci un'immagine del gioco più concreta e precisa ricorrendo alla "sceneggiatura" del gioco stesso, inventandoci, in altre parole, le azioni da compiere affinchè il gioco sia sufficientemente complesso e divertente.

Il nostro omino (sprite) si trova davanti a una riga di labirinto da cui è facilissimo uscire. Ma ecco che, appena l'abbiamo condotto all'uscita, il labirinto si ingrandisce: viene aggiunta un'altra riga e l'omino torna all'inizio, costretto stavolta a percorrere due righe di labirinto.

Il nostro povero sprite, che si illude ogni volta di uscire da quella gran confusione di muri, vede crescere davanti a sè un percorso sempre più grande e, sebbene demoralizzato, lo ripercorrerà tutto dall'inizio. La sua perseveranza verrà premiata se riuscira ad uscire da ben 20 righe. A questo punto il gioco finirà e il computer si complimenterà con lui, anzi con noi, perchè l'abbiamo guidato.

Due ostacoli rendono difficile la vita del nostro omino: i muri elettrici (un miliardo di volts!) che lo fanno tornare alla posizione iniziale se vengono toccati e il tempo limite (5 minuti). Se il tempo a disposizione termina prima del completamento del percorso la partita termina.

Una volta stabilita la "storia" (forse non molto fantasiosa, ma utile per iniziare), è necessario passare alla pratica. Sappiamo bene che tra il dire e il fare c'è di mezzo il computer, ma vedremo di fare del nostro meglio...

Le prime righe di programma

I programmi pubblicati sono tre: la parte facoltativa (ne parleremo alla fine), il gioco scarno e il gioco finito.

Che cosa si intende per gioco scarno? Partendo dall'idea di un gioco è necessario, per arrivare alla versione definitiva dello stesso, suddividere il lavoro in più tappe o fasi. Non possiamo certo pretendere, all'inizio, di programmare un videogame in un colpo solo!

Il programma scarno è una bozza del gioco. Deve essere molto povero e contenere solo l'essenziale. Solo successivamente, quando tutto funzionerà a dovere, potremo aggiungere colori, grafica, suoni e sofisticarlo ulteriormente arrivando al programma finito.

Ma torniamo a noi: il gioco scarno (d'ora in poi lo chiameremo così) è molto più povero del gioco finito: non è previsto il controllo del tempo, mancano i messaggi di condoglianze (!) e di vittoria, non è possibile la scelta del livello di difficoltà, è privo di suoni e di colori. E' bene che sia così perchè sarà più facile analizzarlo, rintracciare eventuali errori e stabilire se la "storia" è così interessante come sembrava al momento della progettazione. In seguito magari, passo per passo, lo integreremo fino a giungere al terzo programma (gioco finito). Prima, però, chiariamo alcuni punti che serviranno per comprendere meglio il programmino.

Parlar di sprite

Probabilmente già sapete che uno sprite deve essere disegnato su una griglia di 24*21 quadretti (rispettivamente orizzontale - verticale). In ogni fila possiamo distinguere tre "sottofile" (parte di una fila: termine brutto ma significativo) da otto quadretti l'una (24/3=8) per un totale di 63 sottofile (8*21). Ogni sottofila è un byte e ogni quadretto si chiama bit. L'immagine di un qualsiasi sprite occuperà quindi 63 bytes di memoria.

I bit possono trovarsi in due stati, accesi o spenti: i quadretti accesi corrispondono ad un 1 (uno) e quelli spenti a uno zero (0). Ogni byte è quindi un numero binario composto da otto cifre che possono essere zero o uno, come per esempio 01011011, 00011000, 10101010, 11111111, eccetera.

Arrivati a questo punto dobbiamo convertire in decimale i 63 numeri binari che danno forma allo sprite seguendo questa semplice procedura: a ciascuno degli otto bit (cifre) corrisponde un valore decimale.

Partendo da destra, pertanto, i suddetti valori sono: 1, 2, 4, 8, 16, 32, 64, 128. Per ottenere il valore decimale, sommiamo tra loro i numeri corrispondenti ai bit accesi (1). Il numero binario 11000001 per esempio, corrisponde a:

128+64+0+0+0+0+0+1=193 (valore decimale).

I più brillanti di voi avranno senz'altro capito che un numero binario di 8 bit (come nel nostro caso) non può essere minore di zero (00000000) e non può superare il valore 255 (11111111).

Il nostro sprite (che assume il ruolo di omino) deve essere molto piccolo per muoversi agilmente nel labirinto: due pixels (=bit sullo schermo) sono la grandezza ideale. Di tutta la griglia per disegnare gli sprites, noi occupiamo quindi i primi due bit del primo byte, che avrà come valore decimale 192 (128+64). I restanti 62 byte avranno valore nullo (zero) e saranno, come si dice in gergo, "resettati".

Come certo saprete, la memoria del Commodore 64 è formata da tante locazioni (esattamente 65536) numerate da 0 a 65535. Ognuna di queste "contiene" un byte e quindi un numero compreso tra 0 a 255. Sarà proprio in alcune di queste locazioni che depositeremo i 63 numeri che danno forma allo sprite. Attenzione, però: non è possibile partire da una locazione qualsiasi (ad es. la 36547) per memoriazzare i valori, ma devono verificarsi tre condizioni, cioè:

- la locazione iniziale deve essere un multiplo di 64 poichè, per segnalare al computer dove rintracciare i valori relativi al nostro sprite, si deve immettere nella locazione 2040 (che assume il compito di pro-memoria o "puntatore") il numero della prima locazione diviso per 64 (indirizzo = locazione). Rileggete attentamente e lentamente se il concetto non è chiaro;

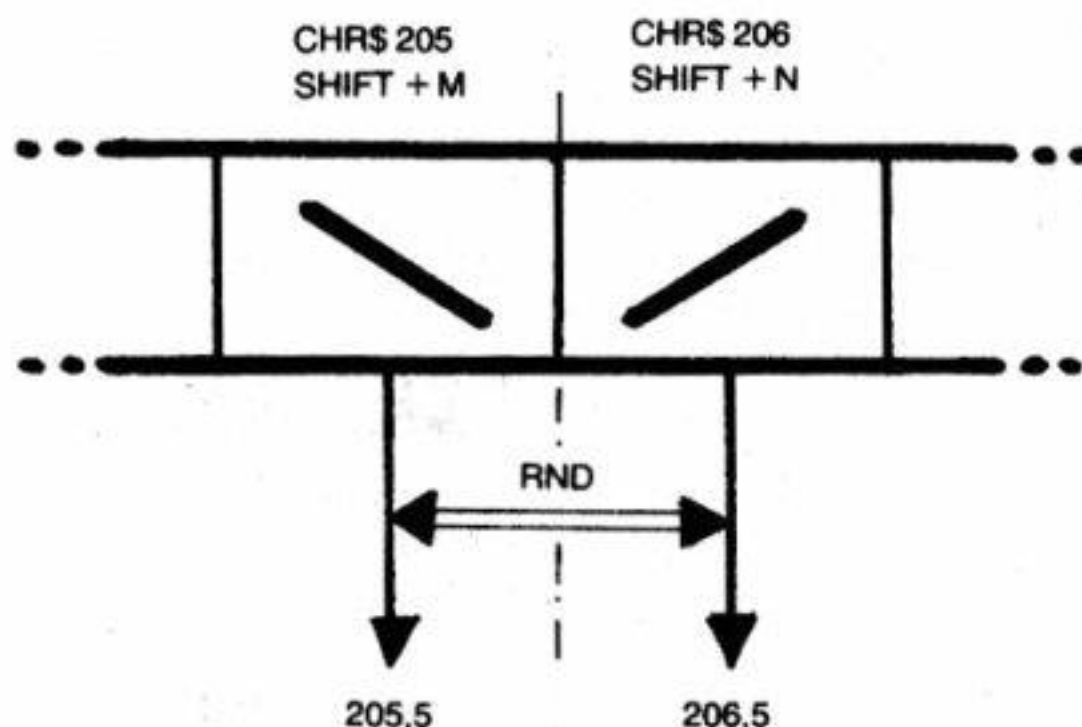



Figura 1 - Sommando a 205,5 un numero casuale compreso fra zero e uno, si ottiene un secondo numero che sarà l'argomento della funzione CHR$. Questa funzione prende in considerazione soltanto la parte intera del numero, quindi si avrà il 50% di possibilità che il numero sia 205 (sbarretta SHIFT + M); altrettante sono le possibilità che il numero sia 206 (codice della sbarretta SHIFT + N).

● la locazione di inizio non può essere superiore a 16320. Questo è il valore massimo che può assumere l'indirizzo della locazione, dato che il puntatore non può essere superiore a 255. Tale valore, moltiplicato per 64, fornisce appunto 16320.

Si potrebbe andare anche oltre questo numero, ma sarebbe necessario ricorrere a tecniche di programmazione piuttosto complesse;

● con i nostri dati non dobbiamo, inoltre, "contaminare" locazioni usate dal sistema operativo del calcolatore.

Per i dati del nostro omino utilizzeremo, in definitiva, le locazioni numerate da 832 a 895 che possiamo considerare libere.

La riga 500 (ci riferiamo al programma scarno, che è il secondo listato) azzera tutti i bytes da 832 ad 895 e pone, mediante istruzione POKE, nella locazione 2040 (puntatore del nostro sprite) il valore 13. Infatti 832 diviso 64 fa proprio 13. La figura 2 dovrebbe spazzare via ogni ombra di dubbio su quanto è stato detto finora.

Nella riga successiva viene immesso in locazione 832 il valore 192 (due pixels affiancati). Quando uno sprite è composto da pochi punti, come nel nostro caso, è superfluo usare il metodo tradizionale per immettere i valori in memoria (READ & DATA): è molto più breve, più veloce e più comodo azzerare i 63 bytes interessati (mediante FOR-...NEXT) e pokare (immettere in memoria) i pochi valori diversi da zero nelle locazioni desiderate.

La riga 515 stampa (visualizza sullo schermo) 27 spazi in reverse (negativi) allineati, per ottenere un "muro" che impedisca al nostro sprite di uscire dallo schermo verso l'alto. La riga successiva crea una zona libera, sbarrata ai lati con due spazi reverse; il centro di questa riga libera sarà il punto di partenza dell'omino.

La riga 518 controlla, per mezzo di un'istruzione IF...THEN se le righe di labirinto "coperte" sono 20 e, se la condizione si avvera, pone fine al gioco (END).

La riga 520 stampa uno spazio in reverse, che sarà la barriera laterale sinistra della riga di labirinto che verrà stampata a lato.

Nella riga 530 viene calcolato un numero casuale compreso fra zero e due per mezzo della funzione RND(1); essa genera un numero decimale casuale compreso fra zero e uno (diverso, però, da 0 e 1). Il numero casuale viene poi moltiplicato per il contenuto della variabile LV, alla quale era stato precedentemente assegnato il valore 3 (riga 505). Nella variabile NU viene infine posta la parte intera del risultato, che sarà maggiore di 0 (cioè 1) e minore o uguale a 2 (cioè ancora 1 oppure 2).

Perchè questo? Supponiamo che la funzione RND generi il numero più alto, cioè 0,999... Questo valore viene moltiplicato per tre, ma non si ottiene tre, bensì, ovviamente, 2,999... Di questo numero viene presa la parte intera, cioè 2 (due).

Da ciò possiamo ricavare la seguente regola:
per ogni valore assegnato alla variabile "Y", il calcolo:

X=INT(RND(1)*Y)

porterà come risultato a un valore "X" tale che X sarà comunque maggiore di zero e minore o al massimo uguale a Y-1.

Nelle due righe successive (540-550) viene stampata una riga di labirinto.

La scelta casuale fra le due sbarrette è molto particolare. Ogni carattere o funzione esprimibile con un carattere (es. CLR/HOME = cuore in reverse) sono contrassegnati da un numero detto codice ASCII. Il codice ASCII di tutti i caratteri si trova nelle appendici di qualsiasi manuale Commodore. Il comando PRINT CHR$(X) visualizza il carattere il cui codice ASCII è X.

Che relazione ha tutto ciò con il nostro labirinto?

Il codice CHR$ (ASCII) della sbarretta che si ottiene premendo SHIFT contemporaneamente a M è 205; quello dell'altra sbarretta è 206 (SHIFT + N). Sommando al numero 205.5 un numero casuale compreso fra zero e uno, si ottiene un secondo numero (minimo: 205.5+0=205.5 massimo: 205.5+1=206.5) che sarà l'argomento (valore tra parentesi) della funzione CHR$. La suddetta funzione prende in considerazione, come già detto, soltanto la parte intera del numero e quindi si avrà il 50% di probabilità che il numero

sia 205 (ASCII della sbarretta SHIFT+M); altrettante sono le possibilità che il numero sia 206, codice della sbarretta SHIFT + N (figura 1).

Tra una sbarretta e la successiva viene posto un certo numero di spazi vuoti per rendere più facile la risoluzione del labirinto. Per inserirli viene usata la funzione SPC (derivante da SPaCe) il cui argomento è la variabile NU, cui era stato assegnato un altro numero casuale. Variando il valore della variabile LV avremo quindi un labirinto più difficile (se LV è basso) o più semplice (LV alto).

Un paio di caratteristiche particolari vanno notate a proposito della funzione SPC():

- questa funzione, dopo aver stampato gli spazi, non implica il ritorno a capo anche se il punto e virgola (;) non è presente. Scrivere, per esempio, PRINTSPC(10) (senza punto e virgola) provoca il medesimo effetto di PRINTSPC(10); (col punto e virgola);
- è improprio affermare che questa funzione stampa un certo numero di spazi, perchè, in realtà, provoca semplici spostamenti a destra del cursore.

Al termine della riga di labirinto (sbarrette e spazi) viene stampato un altro spazio in reverse per impedire l'uscita del nostro omino verso destra.

Una visione globale del gioco è illustrata in figura 5.

Una prima limitazione

Molti lettori si saranno posti una domanda: perchè il labirinto è largo solo 25 colonne e non sfrutta l'intero schermo? Il video del C-64 è composto da 40 colonne e, dato che in ciascuna di queste può prender posto un carattere di formato 8x8 pixel (puntini elementari), il numero di pixel orizzontali è 320.

Uno sprite non può andare oltre i tre quarti dello schermo (ai quali corrisponde la massima coordinata orizzontale che è 255) perchè nella locazione destinata a contenere l'ordinata è possibile memorizzare un numero compreso tra zero e

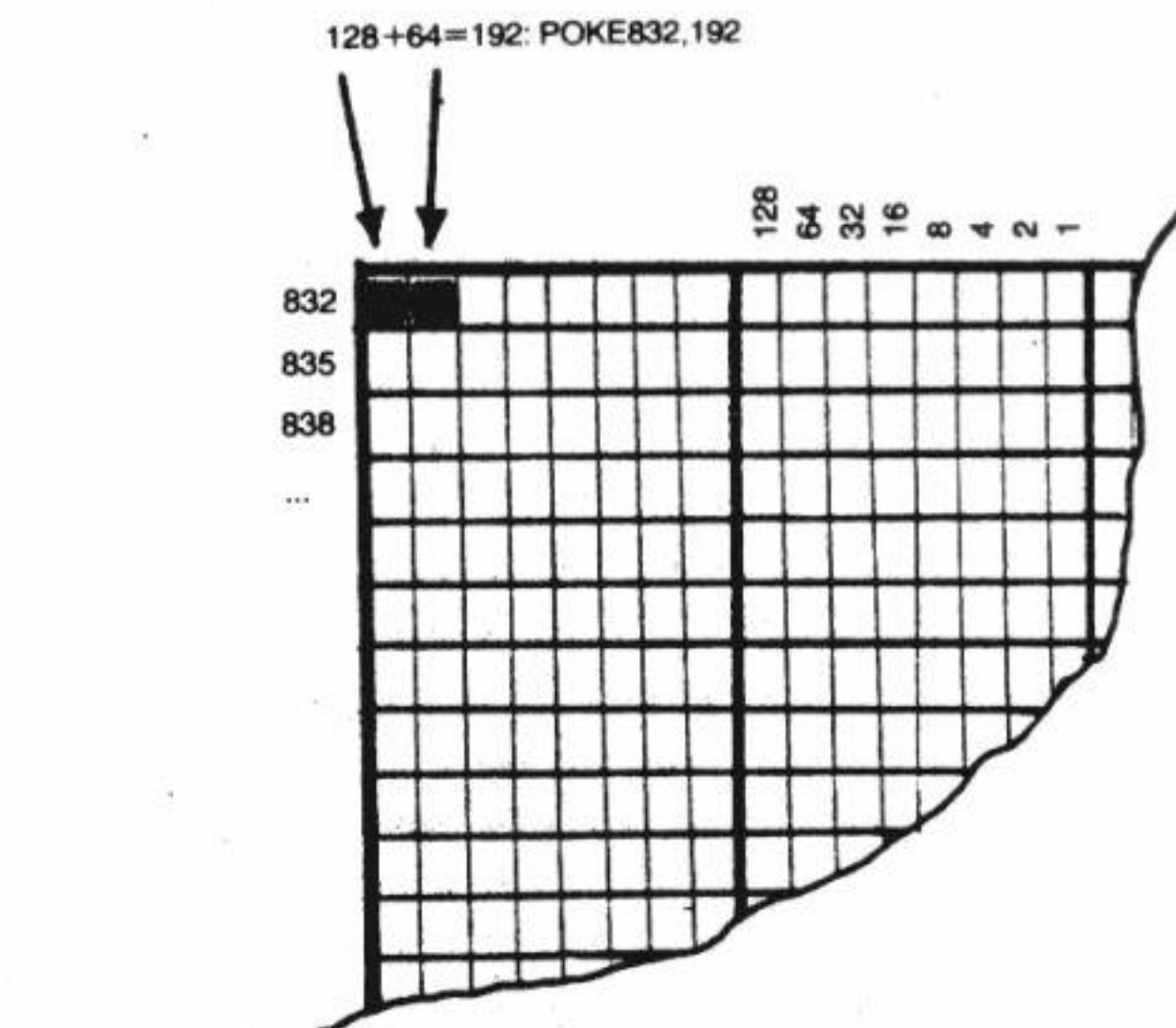



Figura 2 - La figura rappresenta l'angolo in alto a sinistra della griglia 24*21 che compone il nostro sprite. Questo è formato da due soli pixel.

255 (tra poco parlemo di come posizionare uno sprite).

Sarebbe possibile ovviare a tale inconveniente, ma in questa sede non è opportuno dire come, per non complicare il discorso.

Siamo cos- arrivati alla riga 560 che contiene una POKEV+21,1. Questa istruzione abilita lo sprite, facendolo comparire sul video.

V+21 (V=53248, riga 500) significa 53248+21, cioè 53269, locazione che controlla l'abilitazione (tacceso, spento) degli sprites.

Perchè scrivere allora POKEV+21,1 e non direttamente POKE53269,1? Perchè tutte le locazioni da 53248 a 53294 hanno a che fare con gli sprites (coordinate, abilitazione, espansione, colore, priorità, collisioni eccetera). Ricordarsele tutte è un po' arduo, vi pare? E' molto più semplice ricordare un numero di due cifre (tra cui 21) e sommarlo a una variabile (V) cui è stato precedentemente assegnato il valore 53248.

Torniamo al programma scarno.

La riga 570 pone nella variabile X il valore 134 e nella variabile Y il valore 60. Queste due variabili contengono le coordinate dello sprite (X ascissa Y ordinata). 134 e 60 sono le coordinate del nostro omino in posizione di partenza, ovvero nella parte alta dello schermo. Rimandiamo sempre alle illustrazioni per chiarire il discorso (figura 5).

La riga successiva (580) immette nella locazione di memoria V (53248) il valore di X e nella locazione V+1 il valore di Y. La locazione 53248 contiene la coordinata orizzontale (X) dove lo sprite deve essere visualizzato, mentre la locazione 53249 contiene la coordinata verticale (Y = ordinata).

Le coordinante partono dal lato sinistro dello schermo per la X e dal lato superiore per la Y. Il punto definito con tali valori, distante X da sinistra e Y dall'alto, coincide con l'angolo in alto a sinistra (anche se spento) della griglia usata per disegnare lo sprite. Per la precisione è bene ricordare che le due distanze non vengono prese dai bordi della parte visibile dello schermo, ma da un rettangolo più ampio comprendente anche la zona

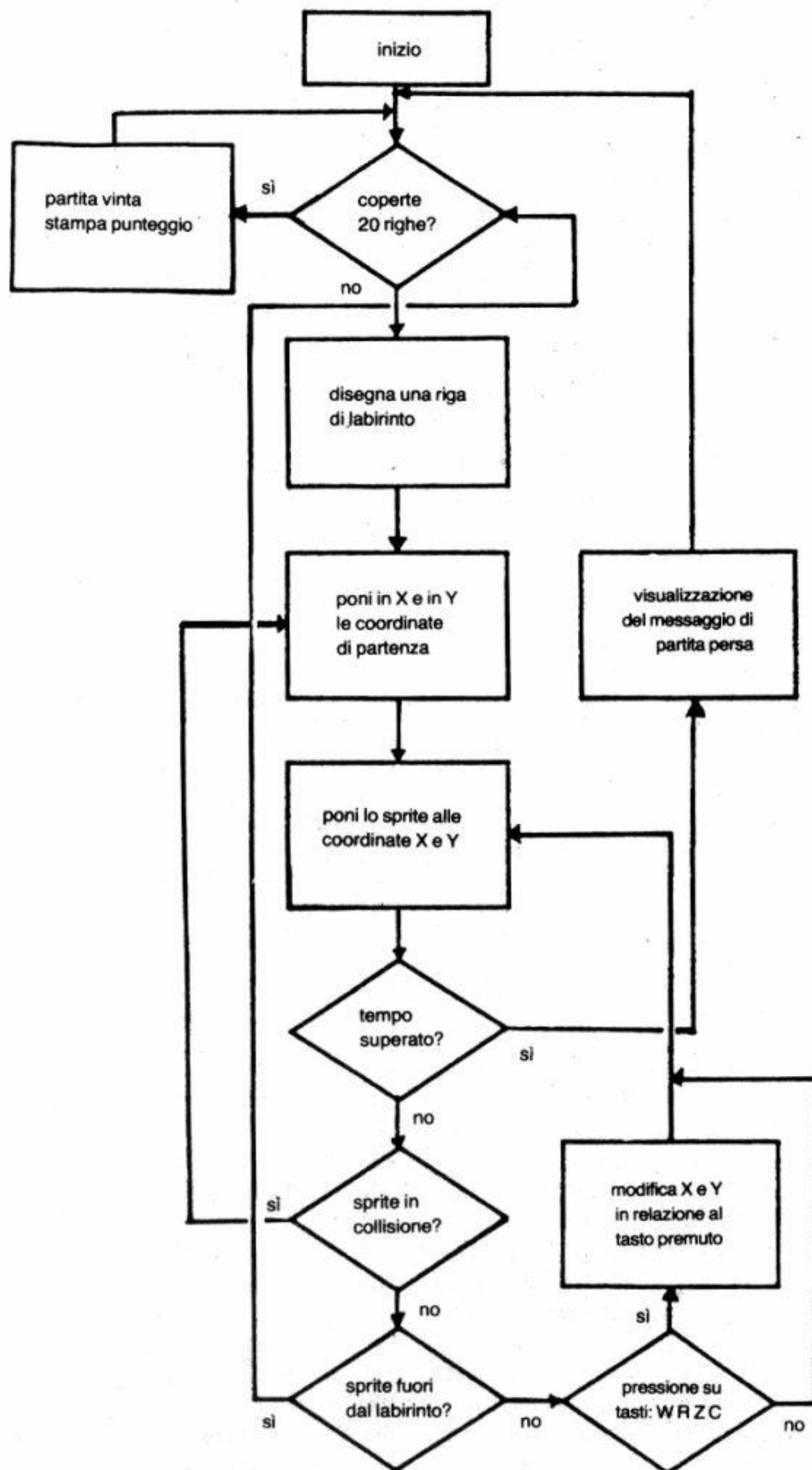



Figura 3 - Diagramma di flusso del videogioco finito (terzo listato). I rombi indicano un salto condizionato, cioè un'istruzione del tipo IF...THEN.

del bordo del video. Questo fatto può esser sfruttato, ad esempio, per far comparire uno sprite all'improvviso, dall'esterno.

La gestione delle collisioni

Passiamo alla riga 585, trascurando, per il momento, il resto della riga 580.

Viene letto (PEEK) il contenuto della locazione V+31. La locazione V+31 rivela un'eventuale collisione tra uno sprite e un carattere qualunque presente sullo ???? dello schermo (come quelli che formano i muri del labirinto). Se il nostro sprite (numerato con zero) "tocca" un carattere o un qualsiasi punto acceso dello schermo (che non sia un altro sprite), il bit più a destra di questa locazione diventa 1 lasciando inalterati gli altri e, di conseguenza, anche il contenuto della locazione è 1, in decimale.

La locazione V+31 ha delle particolarità come del resto anche la locazione di collisione fra sprite, che però non compare nel gioco:

- dopo una lettura (PEEK), la locazione si azzera automaticamente. Se lo sprite che era in collisione lo è ancora (dopo la lettura), V+31 riprende il valore 1;
- se lo sprite è in collisione con un carattere, la locazione in questione assume il valore 1. Se poi viene tolto dallo stato di collisione, modificando opportunamente le coordinate, la locazione mantiene, nonstante ciò, il valore 1. Deriva, da ciò, una fondamentale considerazione:

la locazione V+31 si azzera solo dopo una lettura con un'istruzione PEEK.

Ora siamo in grado di capire cosa succede nel programma.

Se lo sprite è in collisione, PEEK(V+31)=1, ovvero il nostro omino ha tòccato un muro, si torna alla riga 570 dove le coordinate vengono poste a 134 e 60 (omino in partenza). Se lo sprite non è in collisione si passa alla riga successiva.

Precedentemente si è rinviata la descrizione dell'ultima parte della riga 580

che mette nella variabile J il contenuto della locazione V+31 (ora sappiamo a che cosa serve). Questa variabile, J, non verrà tuttavia utilizzata nel programma. A che serve, allora?

Supponiamo che si sia verificata una collisione e vediamo in sequenza cosa avviene:

- la riga 585 rileva la collisione;
- si passa alle istruzioni successive a THEN;
- il contenuto di V+31 è ancora 1 perchè lo sprite è sempre in collisione;
- si torna alla riga 570;
- lo sprite viene rimesso nella posizione iniziale, tuttavia il contenuto di V+31 è ancora 1, per i motivi detti prima;
- dopo aver posizionato lo sprite, la locazione V+31 viene letta per azzerarla, assegnandone il contenuto ad una variabile a caso (J). In questo modo la locazione delle collisioni, per il semplice fatto di esser stata letta, torna a zero.

Le cose possono andare anche diversamente. Può succedere infatti che lo sprite venga messo in stato di collisione (riga 580). Successivamente viene letto il contenuto di V+31 (con J=PEEK...) che è ovviamente 1 (uno). V+31 si azzera (è avvenuta una lettura), ma torna subito a 1 perchè lo sprite è in collisione. Se però l'istruzione IF...THEN della riga 585 legge il contenuto di V+31 prima che il computer l'abbia rimesso a 1, può avvenire che il programma non rilevi la collisione. Di conseguenza il nostro omino potrebbe trapassare tranquillamente i muri.

Ricorrendo invece a un ciclo di ritardo (FORI=1TO10) molto breve, ma sufficiente perchè il sistema operativo ponga in V+31 il valore 1 (parte finale della riga 580) si risolve il problema.

Se non avete capito tutto questo discorso non scorraggiatevi e rileggete quest'ultima parte dell'articolo.

Siamo così giunti, nel nostro lungo cammino, alla riga 590. Un'istruzione IF...THEN verifica se Y (coordinata verticale dello sprite) è maggiore di 75+8*DI. DI contiene il numero di righe di labirinto "risolte". L'espressione matematica sopracitata dà come risultato la coordinata verticale in cui termina il labirinto. Se la condizione si attua, vuol dire che l'omino si trova fuori dal labirinto. Il valore di "DI" viene incrementato di un'unità e il controllo va (GOTO) alla riga 518. Qui viene verificato il valore di DI: se DI è maggiore o uguale a 20, il nostro omino è uscito da 20 righe di labirinto e la partita termina (END). In caso contrario il labirinto viene allungato di una riga.

Le righe 600, 610, 620 e 630 sono addette al controllo dei tasti. Probabilmente non conoscete la locazione 197: contiene in ogni istante il codice, completamente differente dal codica ASCII, del tasto premuto. Se (IF) il contenuto della locazione 197 è uguale al codice del tasto che si vuole sia premuto perchè la condizione si avveri, allora (THEN) il programma modifica opportunamente le coordinate e torna alla riga 580.

Se volete conoscere il codice 197 (designamolo pure così) di un qualsiasi tasto fate girare a parte la seguente microriga e premente un tasto: sul video comparirà il suo codice:

```
1 PRINTPEEK(197):GOTO1
```

Ma torniamo al controllo della tastiera.

Se nessun tasto viene premuto si torna (GOTO di riga 640) al primo IF...THEN addetto al controllo tasti, ovvero alla riga 600.

Avete sicuramente notato che i corridoi del labirinto sono in posizione obliqua (45 gradi) rispetto allo schermo; per comodità quindi, anche l'omino si dôvrà muovere in diagonale e non in orizzontale - verticale. Questo si ottiene variando opportunamente le coordinate: per la direzione alto-sinistra, per esempio, viene decrementata di uno sia X che Y per i motivi che ben potete immaginare. Anche i tasti devono essere disposti in diagonale, ecco giustificata la scelta di W, R, Z, C.

Siamo giunti al termine, finalmente.

Il gioco vi sembra brutto? Allora al lavoro: aggiungiamo colori, suono, tempo, punteggio e livello di difficoltà.

Come migliorare il programma

Se avete registrato su nastro o disco il programma scarno, caricatelo ora in memoria. Per "trasformarlo" nel programma finito (terzo listato) non è, infatti, necessario ribatterlo per intero, ma sarà sufficiente aggiungere, cancellare o modificare alcune righe del listato precedente. Questo, anzi, può essere un otti-

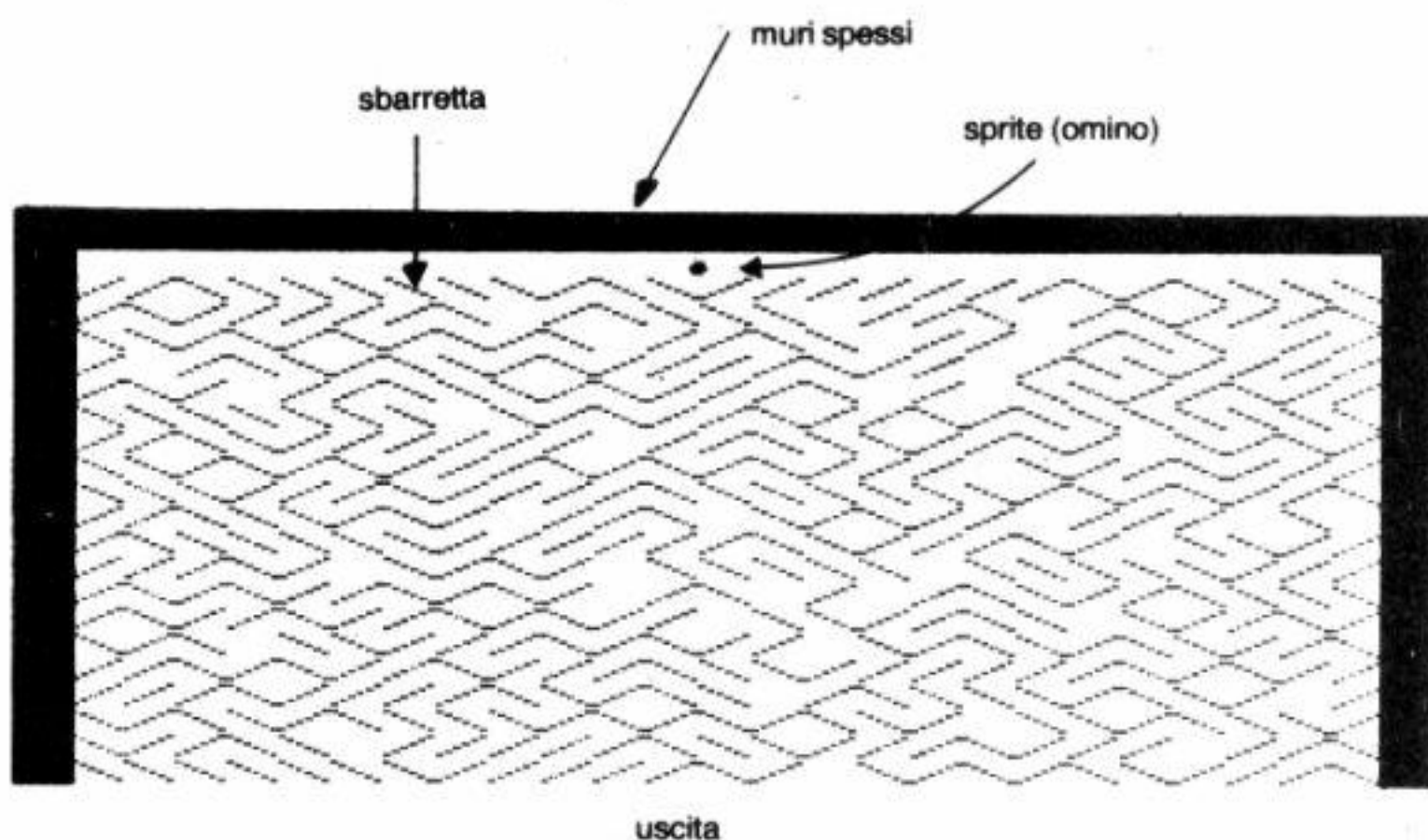



Figura 4 - Ecco come appare il gioco sul video. Il puntino in alto è l'omino-sprite che dobbiamo condurre fuori dal labirinto. I muri spessi impediscono allo sprite di andare troppo in alto (fuori dallo schermo) o di uscire lateralmente.

mo esercizio per individuare le migliorie apportate.

Se avete eseguito tutto correttamente, fate girare il gioco: è o non è più gradevole?

I suoni

Il Commodore 64 è dotato di un circuito sonoro molto potente e versatile, ma relativamente complesso da programmare. Non abbiamo intenzione di scrivere un trattato su questo argomento sia perche CCC ne ha già parlato (e ne parlerà ancora...), sia perchè non è tutto così facile.

Il circuito sonoro (SID) può suonare fino a 3 voci separate o contemporaneamente. Per semplicità ci riferiremo sempre alla voce N.1 invitando quanti vogliono approfondire a consultare il manuale del 64 o i numeri arretrati di CCC.

Come per gli sprites, le locazioni (o registri) che controllano il SID si trovano in un unico "blocco", numerato da 54272 fino a 54300. Scriveremo quindi nel programma S=54272; ogni successivo POKaggio in memoria che avrà a che fare con i suoni seguirà la sintassi POKES+XXX,YYY.

Vediamo ora di mettere in luce come è strutturato un suono.

Vi sarà capitato di premere il tasto di un pianoforte e di ascoltarne l'emissione sonora:

- il volume del suono sale rapidamente al valore massimo;
- successivamente si attenua;
- persiste fino all'istante in cui togliete il dito dal tasto;
- poi scende a zero rapidamente.

Per poter riprodurre lo stesso effetto sonoro col calcolatore dovrete definire i quattro valori delle fasi di sviluppo del suono che si susseguono così:

"A": tempo in cui il suono raggiunge il suo massimo volume;

"D": tempo in cui il suono si porta al volume medio di persistenza;

"S": tempo in cui il suono continua a rimanere attivo;

"R": tempo in cui il volume si azzera.

In inglese i termini corrispondenti alle quattro lettere ADSR sono i seguenti: Attack, Decay, Sustain, Release.

Questi valori devono essere immessi nelle locazioni S+5 e S+6 e variano da 0, tempo brevissimo a 15, lungo.

Il registro S+5 viene diviso in due parti di 4 bit: la parte alta viene usata per memorizzare il valore dell'attack, quella bassa per memorizzare la durata del decay.

Anche il registro S+6 viene scisso in due: parte alta = sustain, parte bassa = release.

Poichè 4 bit possono rappresentare valori da 0 a 15, avete la possibilità di scegliere un valore per ognuna delle 4 fasi in questo intervallo. Tuttavia per memorizzare "A", una volta definito il suo valore, nella parte alta del registro 5, bisognerà ricordarsi di eseguire il prodotto per 16. La medesima operazione va effettuata con S ed il registro St6.

Per i valori di D e di R non vanno eseguiti calcoli, essi vanno rispettivamente sommati ai valori di D e di S.

Le locazioni S e S+1 devono contenere i due numeri che determinano la frequenza della nota (byte basso e alto); le frequenze si trovano nelle appendici del manuale Commodore.

Il registro S+24 contiene il volume (unico per le tre voci), da 0 (spento) a 15 (alto).

Una volta pokati nelle locazioni sopra descritte i valori desiderati, possiamo dare il via al suono con una POKES+4,17 (forma d'onda "dolce"). Per dare inizio alla fase R (release) si impartisce: POKES+4,16 (cioè 17-1).

Nel programma, la linea 508 genera un breve suono di frequenza casuale: ogni volta che lo desideriamo, eseguiamo GOSUB508 (alla pressione di un tasto, quando l'omino entra in collisione, ecc.).

Per concludere questo argomento, ricordatevi che nelle prime righe di un qualsiasi programma che utilizza i suoni deve essere presente una routine che resetta (mette a 0) tutti i registri del SID come ad esempio:

```
FORI=0TO24:POKES+I,0:NEXT
```

La variabile di tempo

Se avete fatto girare il gioco, avrete notato che il tempo viene visualizzato alla destra dell'ultima riga di labirinto. Quando ne viene aggiunta un'altra, il tempo si sposta più in basso e sopra resta il tempo impiegato per risolvere il labirinto fino a quel punto. Per visualizzare il tempo e controllare se è scaduto (righe 582 e 583) anche quando non sono premuti tasti, abbiamo dovuto modificare la struttura del programma come si vede nella figura 3. Il controllo del tempo, che ha bisogno di qualche istante per essere eseguito può ora sostituire il breve ciclo di ritardo che era presente nella riga 580 del programma scarno.

La variabile speciale TI$ contiene sempre l'ora e parte da 0 all'accensione del computer, che provvede a incrementarla. Noi possiamo modificarla: ad esempio, se sono le nove e mezza e trentadue secondi, scriveremo:
TI$="093032".

All'inizio della partita (riga 517), TI$ viene posta a 0 ("000000"); la riga 583 controlla se sono passati 5 minuti (TI$>="000500") e, in questo caso, la partita termina con missione incompiuta e viene visualizzato il punteggio.

Altre modifiche

Un abbellimento importante è quello cromatico: sono stati inseriti i colori nelle istruzioni PRINTCHR$(). In riga 513 vengono variati il colore dello schermo (locazione 53281) e il colore del bordo (locazione 53280): ad ogni numero corrisponde un colore.

Ricordate la variabile LV, dal cui valore dipende la difficoltà del labirinto? Con un'istruzione INPUT (riga 514) si è inserita nel programma la scelta del livello di difficoltà, da 1 (difficile) a 5 (facile). Si possono introdurre anche valori decimali (per es. 2.4); se viene immesso un numero minore di 1 o maggiore di 5, LV viene posta automaticamente a 3 (piuttosto facile).

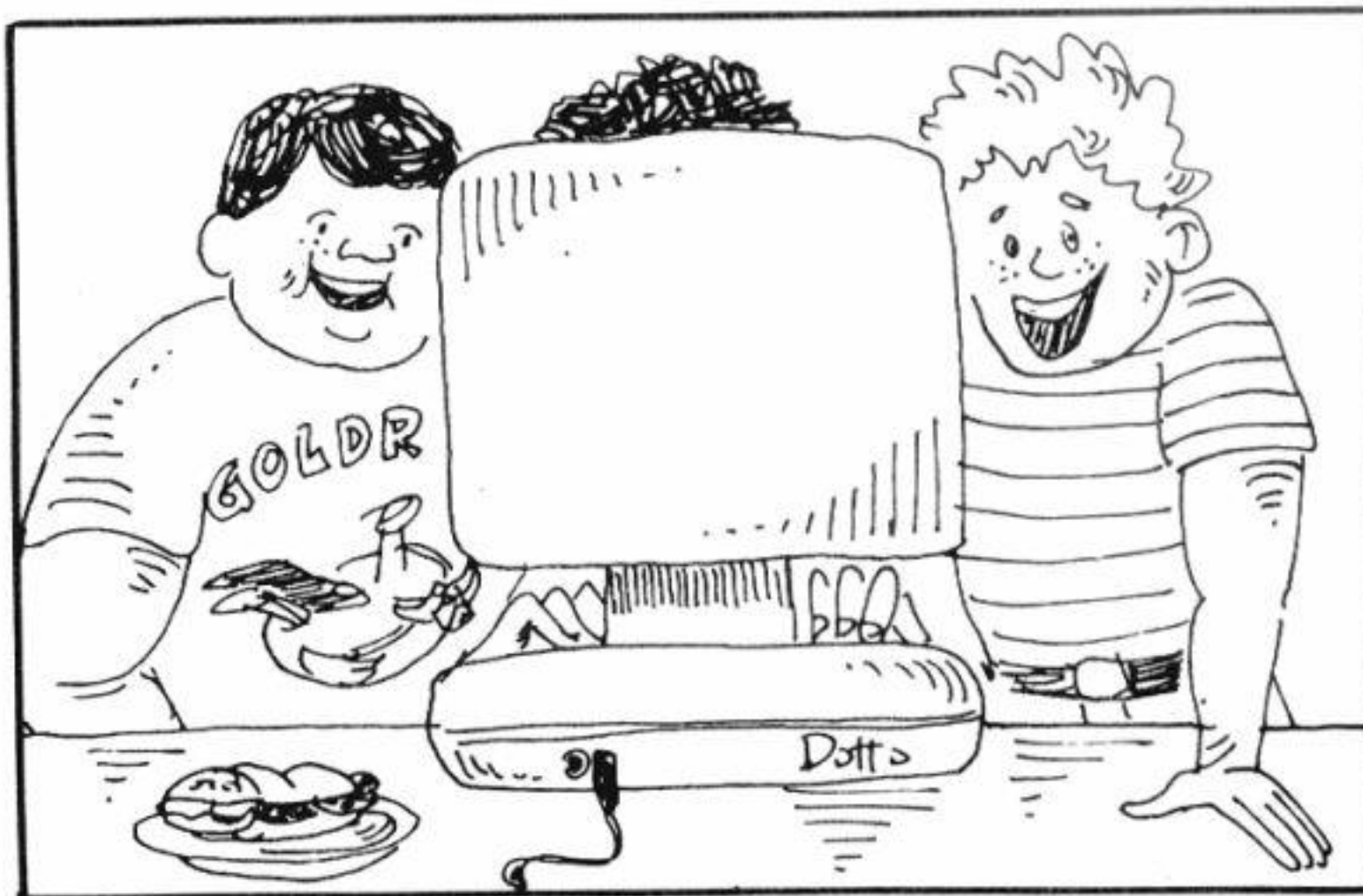
Il punteggio finale, tramite un'espressione matematica (riga 1040), tiene conto delle righe di labirinto portate a termine, del tempo impiegato e del livello di difficoltà scelto (più difficile = più punti). In caso di missione compiuta (20 righe) viene aggiunto un bonus di 500 punti.

```
1 A$(0)="\":A$(1)="/":A$(2)=CH
  R$(32):PRINTA$(RND(0)*10/3);
  :GOTO 1

100 REM  IMPARIAMO A PROGRAMMARE
110 REM  GLI SPRITE COL C-64
115 :
500 FOR I=832 TO 895:POKE I,0:NE
    XT:POKE 2040,13:V=53248
505 POKE 832,192:LV=3
510 PRINTCHR$(147); TAB(30)"W R
    Z C"
515 PRINTCHR$(19);CHR$(18)"
                          ":REM
     27 SPAZI
517 PRINTCHR$(18);CHR$(32); TAB(
    26);CHR$(32)
518 IF DI=20 THEN POKE 198,0:END

520 PRINTCHR$(18);CHR$(32);CHR$(
    146);:FL=0
530 NU=INT(RND(1)*LV):FL=FL+NU+1
540 IF FL<25 THEN PRINTCHR$(205.
    5+RND(1));SPC(NU):GOTO 530
550 PRINT TAB(25);CHR$(205.5+RND
    (1));CHR$(18)CHR$(32)
560 POKE V+21,1
570 X=134:Y=60
580 POKE V,X:POKE V+1,Y:J=PEEK(V
    +31):FOR I=1 TO 10:NEXT
585 IF PEEK(V+31)=1 THEN FOR I=1
     TO 500:NEXT:GOTO 570
590 IF Y>=75+8*DI THEN DI=DI+1:G
    OTO 518
600 IF PEEK(197)=12 THEN X=X-1:Y
    =Y+1:GOTO 580
610 IF PEEK(197)=20 THEN X=X+1:Y
    =Y+1:GOTO 580
620 IF PEEK(197)=9 THEN X=X-1:Y=
    Y-1:GOTO 580
630 IF PEEK(197)=17 THEN X=X+1:Y
    =Y-1:GOTO 580
640 GOTO 600

100 REM  IMPARIAMO A
110 REM  SOFISTICARE
120 REM  I PROGRAMMI
130 :
500 FOR I=832 TO 895:POKE I,0:NE
    XT:POKE 2040,13:V=53248:S=54
    272
505 POKE 832,192
506 FOR I=0 TO 24:POKE S+I,0:NEX
    T:POKE S+6,240:POKE S+24,15:
    POKE S,40:GOTO 513
508 POKE S+1,50*RND(1):POKE S+4,
    17:FOR ZX=0 TO 30:NEXT:POKE
    S+4,16:RETURN
513 POKE 53281,11:POKE 53280,2:P
    RINTCHR$(147)CHR$(159);CHR$(
    17);
514 INPUT "LIVELLO DI DIFFICOLTA
    ' 1(D) 5(F)";LV:IF (LV<1) OR
     (LV>5) THEN LV=3
515 GOSUB 508:PRINTCHR$(147);CHR
    $(158);CHR$(18);
516 GOSUB 508:PRINT"TASTI:-W---R
    ---Z---C-------";CHR$(146);"
    -TEMPO:-"
517 PRINTCHR$(18);CHR$(32); TAB(
    26);CHR$(32):TI$="000000"
518 IF DI=20 THEN A$=TI$:DI=DI-1
    :GOTO 1000
520 PRINTCHR$(18);CHR$(32);CHR$(
    146);:FL=0
530 NU=INT(RND(1)*LV):FL=FL+NU+1
540 IF FL<25 THEN PRINTCHR$(205.
    5+RND(1));SPC(NU):GOTO 530
550 PRINT TAB(25);CHR$(205.5+RND
    (1));CHR$(18)CHR$(32)CHR$(14
    6);
560 POKE V+21,1
570 X=INT(RND(1)*240):Y=60
580 POKE V,X:POKE V+1,Y:J=PEEK(V
    +31)
582 PRINT TAB(30);RIGHT$(TI$,4);
    CHR$(145)
583 IF TI$>="000500" THEN 2000
585 IF PEEK(V+31)=1 THEN GOSUB 5
    08:FOR I=1 TO 500:NEXT:GOTO
    570
590 IF Y>=75+8*DI THEN DI=DI+1:P
    RINT:FOR G=1 TO 10:GOSUB 508
    :NEXT:GOTO 518
600 IF PEEK(197)=12 THEN X=X-1:Y
    =Y+1:GOTO 580
610 IF PEEK(197)=20 THEN X=X+1:Y
```

```
     =Y+1:GOTO 580
 620 IF PEEK(197)=9 THEN X=X-1:Y=
     Y-1:GOTO 580
 630 IF PEEK(197)=17 THEN X=X+1:Y
     =Y-1:GOTO 580
 640 GOTO 580
1000 PRINTCHR$(147)
1010 PRINT" ECCEZIONALE!!! SEI US
     CITO DAL"
1020 PRINT" L A B I R I N T O!!!!
     "
1030 PRINT:PRINT:PRINT:BN=500
1040 PRINT"..PUNTI:+";600-VAL(A$)
     +30*DI+BN-LV*100:PRINT:PRINT
1050 PRINT" ANCORA? (PREMI UN TAS
     TO)":POKE 198,0:WAIT 198,1:G
     OSUB 508:POKE 198,0: RUN
2000 A$=TI$:GOSUB 508:PRINTCHR$(1
     47)
2010 PRINT" IL TEMPO E' TERMINATO
     "
2020 PRINT" RESTERAI PER SEMPRE N
     EL LABIRINTO":PRINT:PRINT:PR
     INT:GOTO 1040
3000 REM  BY MAURIZIO DELL'ABATE
3100 REM  SOLO PER C64 - TASTI W
     Z R C
```

Giochiamo in borsa

Un gioco-simulazione per tre o più partecipanti. Un passatempo "intelligente" per sfruttare al meglio le possibilità del tuo Commodore.

di Flavio Molinari

Giocare in borsa, si dice proprio così: nonostante il giro d'affari di migliaia di miliardi (tra cui, forse, un po' dei nostri soldi, direttamente o indirettamente), nel linguaggio comune si associa al termine "borsa" la parola "gioco".

In effetti gli ingredienti del gioco ci sono tutti: sono necessari abilità, fortuna e piacere del rischio, stando all'immagine offerta da certi film sul mondo finanziario.

Nella pratica le cose vanno un po' diversamente. Oltre alle doti menzionate, sono indispensabili denaro, (molto... si sa: soldi chiamano soldi), grande preparazione su di tutto un po': economia, politica, mondo del lavoro e altro. Infine, come per ogni gioco d'azzardo che si rispetti, bisogna anche saper barare, l'importante è non farsi scoprire... (non prendetela come un'istigazione a delinquere: sono solo banali considerazioni che saltano alla mente sfogliando qua e là i giornali).

Giocare simulando

Da sempre l'uomo ha desiderato ricoprire ruoli diversi da quelli offerti dalla vita quotidiana e in genere le proprie ambizioni sono rivolte verso quei personaggi cosiddetti "di potere". Da questo bisogno sono nati anche i giochi di simulazione: desiderio di potenza, competizione, piacere di misurarsi in situazioni insolite. Tutte queste componenti si intrecciano e si manifestano in gran parte dei nostri passatempi preferiti.

Alcuni hanno origini antichissime, come i giochi di guerra (gli scacchi ne sono l'esempio più sofisticato), altri sono recenti e il più famoso è forse Monopoli.

L'invenzione del calcolatore ha portato una ventata di nuove idee in questo ramo della scienza ricreativa. Grazie alla capacità di calcolo, è possibile costruire modelli che si avvicinano sempre più alla realtà, con applicazioni oggi diventate insostituibili: simulazioni di volo, previsioni del tempo, economia... e, per finire, giochi, argomento più frivolo, ma per questo non meno importante.

Il programma che vi presentiamo è appunto un esempio di gioco-simulazione che, pur essendo molto semplice e lineare nella sua impostazione, dà una valida indicazione su come costruire un modello matematico e applicarlo a un gioco.

Computer e giochi di società

Con l'espandersi del fenomeno home computer, si è avuta parallelamente la proliferazione di tutta una serie giochi, diversi dai video-games tradizionali, che purtroppo sono sovente una brutta copia dei classici passatempi esistenti prima dell'avvento dell'era informatica.

Ottimi sono i programmi per giocare a scacchi, discreti quelli che simulano il tavolo da biliardo, così i programmi per il poker. Forse è solo un retaggio da cultura preinformatica, ma il nostro parere è che un programma, anche se ben fatto, non potrà mai rendere l'atmosfera di un vero tavolo da gioco: il tappeto verde, il rituale che accompagna la visione delle carte (spillare, in gergo pokeristico), l'aria fumosa e irrespirabile, il bicchiere di liquore... Senza contare che è tutta un'altra cosa giocare con dei soldi veri!

Ritornando a fare una personalissima classifica sui giochi di società, il fondo lo si raggiunge con i programmi del tipo tombola, roulette e simili.

Questi sono quanto di più noioso si possa fare con un computer, dato che tutto il fascino tipico di questi giochi va a farsi benedire! Il ritrovarsi in famiglia, o con gli amici, le battute di chi tiene il banco (di solito il più brillante della combriccola), il contatto fisico con le persone nonchè gli oggetti facenti parte del gioco: bottoni, fagioli, carte... con tutto il disordine e l'allegria che ne derivano.

Basta un po' di fantasia

Il programma Borsa non ha certo la pretesa di essere il top dei giochi per computer, ma dà un'idea di come sfruttarne in maniera intelligente le enormi potenzialità, senza ridurlo al mero ruolo di distributore di carte o di visualizzatore di una scacchiera.

L'idea è questa: prendiamo spunto da una situazione reale, la borsa, e facciamone un modello: compra/vendita di titoli azionari, tendenza delle quotazioni a salire o scendere a seconda della richiesta, andamento del mercato. I giocatori dovranno fare la parte di ipotetiche, quanto improbabili, società finanziarie, mentre al computer è demandato il compito che meglio si addice a una macchina del suo genere: elaborazione dati e aggiornamento del listino. La tastiera, da sola, conferisce tra l'altro quell'aria di "professionalità" necessaria ormai in una qualsiasi attività finanziaria.

Abbiamo pensato che il numero ottimale di partecipanti fosse tre: ci si può dividere in tre squadre e, in generale, quanti più siete, tanto più il gioco sarà movimentato e divertente. Il perchè non è possibile giocare in due squadre sarà detto più avanti.

Spiegazione del gioco

Come già anticipato, ci si deve dividere in tre squadre. Il listino di borsa contiene otto titoli in tutto, un giusto compromesso tra il numero effettivo riscontrabile realmente in borsa e considerazioni di tipo pratico.

La prima fase consiste nella scelta del livello di difficoltà (1 oppure 2), del quale parleremo più avanti, e nella "codifica" dei titoli.

Per meglio chiarirne le modalità e gli scopi, riportiamo un esempio di output che vale anche come verifica di una corretta digitazione del listato.

Codifica dei titoli squadra 1
FIAT ?
COMMODORE ?
ORO ?
SYSTEMS ? (un po' di pubblicità non guasta...)
....

Voi rispondete a ciascuna domanda posta con nomi scelti a piacere e tali, comunque, da confondere in seguito le idee alle squadre avversarie (primo riquadro in alto di figura 1), come ad esempio:

PIPPO
ENEL
ORNELLA
TAVOLO
....

Le altre squadre, naturalmente, non dovranno vedere quanto state scrivendo, altrimenti, come vedremo dopo, sarebbe tutto inutile. Per impedire che ciò avvenga, decidete voi il sistema: si proibisce alle altre squadre, durante questa prima fase, di avvicinarsi a meno di due metri dalla consolle; si cerca di coprire col proprio corpo il video; si digitano in fretta i nomi-codice; si cambia colore al cursore rendendolo eguale al fondo...

Durante le successive fasi normali di compra/vendita (in cui cadono, naturalmente, le proibizioni di cui sopra) se, per ipotesi, volete comprare 1000 azioni della Fiat, batterete:

COMPRI ? PIPPO
QUANTITA' ? 1000

a patto, ovviamente, che abbiate associato ai titoli Fiat il nome PIPPO. Lo scopo di questa operazione è di rendere più ardua l'identificazione dei vostri movimenti da parte degli avversari.

Ciò succede anche nella realtà: apposite società operano per conto terzi, per non rendere pubblico il nome di chi sta sotto determinate manovre.

Naturalmente il programma non vuole essere così ambizioso: stiamo costruendo un gioco e per renderlo più interessante non è il caso di tirare in ballo migliaia di variabili. Due semplici vettori (un valore per ogni titolo) vanno bene allo scopo. Questi sono: "R" e "G".
R() è un interruttore a tre stati:
=0 titolo stazionario;
=1 in ribasso;
=2 in ascesa.
G() è il numero di giorni del perdurare dello stato di crisi o ripresa.

Questi elementari artifizi rendono molto più problematica per gli avversari (e anche per voi!) l'interpretazione del listino di borsa. Per distinguere le variazioni casuali generate dal computer dalle oscillazioni prodotte dalla compra/vendita, sarà necessario un maggior intuito e anche un pizzico di fortuna in più: optate per il livello 1 se volete un gioco razionale, privo di fattori imprevisti, altrimenti per il livello 2 dove sono richieste maggiori abilità e fortuna.

Input/output

Durante la compra/vendita il computer vi chiederà:
COMPRI
VENDI
FINE
e sarà necessario premere i tasti "C", "V" oppure "F", a seconda di ciò che volete fare. A turno ogni squadra si avvicenda sulla tastiera, per un totale massimo di quattro "movimenti" (acquisti o vendite) per volta. E' superfluo precisare che potete comprare solo se disponete del liquido necessario, o vendere se effettivamente avete azioni da vendere. In caso contrario il computer vi avviserà dell'errore e tale messaggio potrà esser di aiuto per le squadre avversarie.

Finite le contrattazioni, il ciclo si ripete.
La conclusione della partita è a vostra discrezione: ad esempio potete stabilire in partenza un limite di tempo: vincerà chi allo scadere dei 90 minuti di gioco possiede il capitale più consistente.

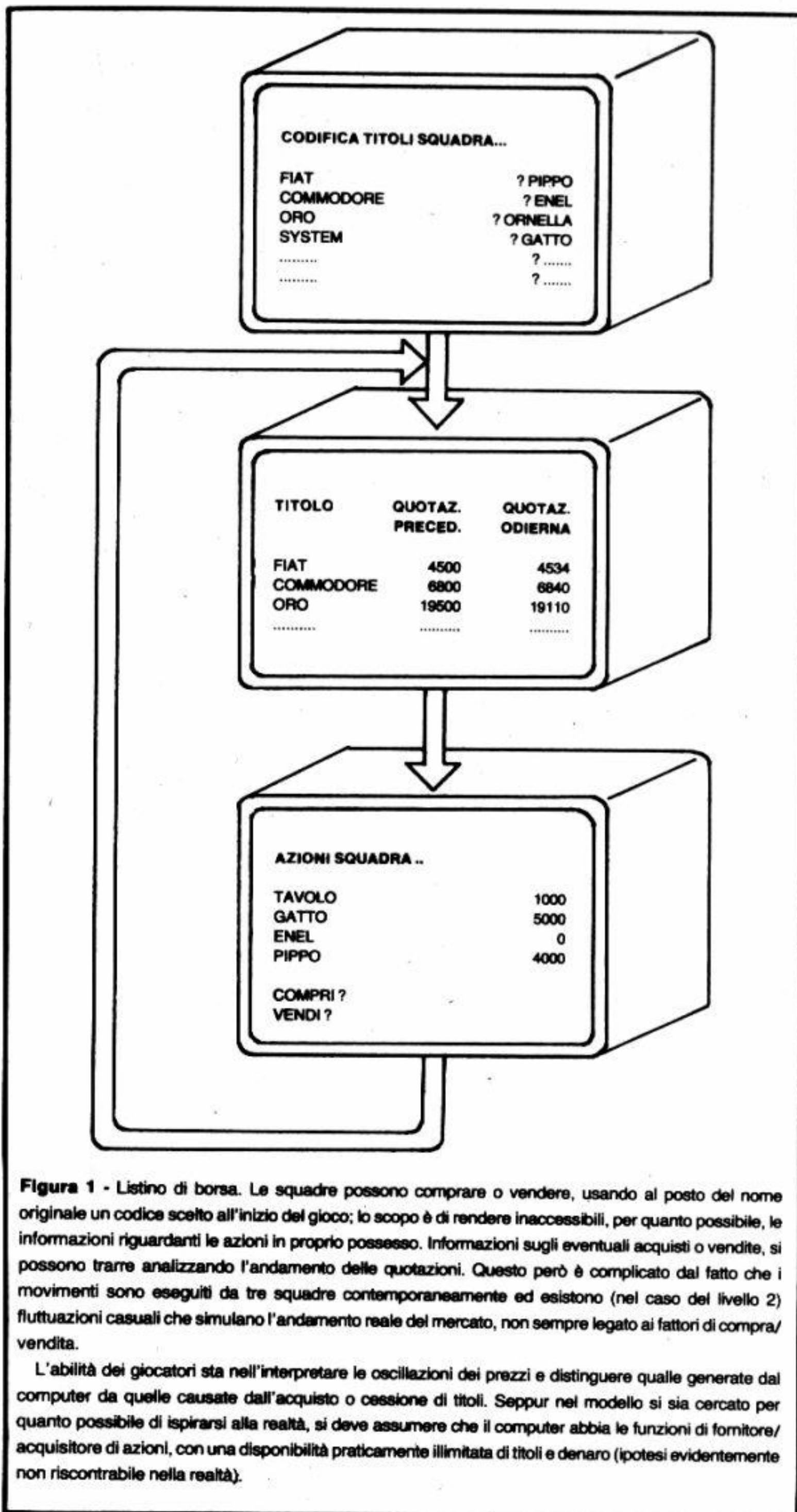



Figura 1 - Listino di borsa. Le squadre possono comprare o vendere, usando al posto del nome originale un codice scelto all'inizio del gioco; lo scopo è di rendere inaccessibili, per quanto possibile, le informazioni riguardanti le azioni in proprio possesso. Informazioni sugli eventuali acquisti o vendite, si possono trarre analizzando l'andamento delle quotazioni. Questo però è complicato dal fatto che i movimenti sono eseguiti da tre squadre contemporaneamente ed esistono (nel caso del livello 2) fluttuazioni casuali che simulano l'andamento reale del mercato, non sempre legato ai fattori di compra/vendita.

L'abilità dei giocatori sta nell'interpretare le oscillazioni dei prezzi e distinguere quelle generate dal computer da quelle causate dall'acquisto o cessione di titoli. Seppur nel modello si sia cercato per quanto possibile di ispirarsi alla realtà, si deve assumere che il computer abbia le funzioni di fornitore/acquisitore di azioni, con una disponibilità praticamente illimitata di titoli e denaro (ipotesi evidentemente non riscontrabile nella realtà).

Figura 2 - Esempio d'uso di una tabella a due dimensioni. Per necessità di gioco, ogni titolo deve essere codificato da ciascuna delle squadre partecipanti (per comodità, è meglio ricorrere a nomi comuni, più facili da ricordare). La corrispondenza codice-titolo è gestita con una matrice a due indici, il primo dei quali punterà al numero del titolo e il secondo al numero della squadra.

Nel programma le "parole" sono memorizzate nel vettore LS$(20,3) le cui dimensioni massime, stabilite in partenza da un'istruzione DIM, sono, al più, 20 titoli (nel programma se ne utilizzano solo 8) per 3 squadre.

Dato che è possibile considerare anche lo zero come indice, nella colonna 0 memorizziamo il nome effettivo del titolo.

Note per gli utenti di Vic-20

Il programma gira senza problemi anche sul più piccolo della famiglia Commodore, purchè dotato di un'espansione di memoria. In un unico caso, nella presentazione del listino, si avrà qualche "sbavatura" nella visualizzazione di alcuni valori. Questo per i noti problemi di dimensioni dello schermo, che per il Vic-20 sono notevolmente ridotti. Intervenite sulla sezione di programma STAMPA LISTINO per far incolonnare in maniera più leggibile i numeri.

Suggerimenti

All'inizio del listato compare una serie di variabili, intervenendo sulle quali si può modificare, anche in maniera sostanziale, l'andamento del gioco.

I valori impostati sono quelli che, dopo alcune prove, risultavano i migliori e che più si avvicinavano a un modello reale. Potete comunque modificarli, al fine di ottenere quello che più si avvicina a vostri gusti personali. Attenzione però a non immettere numeri "sballati", altrimenti vi saranno titoli che impazziscono, con quotazioni che salgono alle stelle o che scendono in picchiata verso lo zero.

Questo è il tipo di intervento più semplice. In un secondo tempo, per dare un pizzico di realismo in più, potete far comparire una schermata contenente brevi note sull'andamento dei titoli (un piccolo notiziario di borsa). Un confronto tra le variabili:

QZ(titolo,0) quotaz. precedente
QZ(titolo,1) quotaz. attuale

fornirà le informazioni necessarie.

Se ancora non siete contenti, ma questo è lavoro per i più esperti, rendete più realistica l'oscillazione delle quotazioni, diversificando l'ampiezza e la durata dei periodi di crisi o ascesa (movimenti del mercato).

Fermiamoci qui. Quello che più conta è la fantasia. Anche saper programmare è importante, ma questo non è un problema: il Basic si impara...

Ricordate il giallo dell'estate scorsa che ha coinvolto le azioni Bi-Invest? Ciò che appassionava e attirava la curiosità del vasto pubblico, consisteva nel fatto che nessuno (?) sapeva chi stava effettivamente rastrellando le azioni della holding milanese. Il malcapitato Carlo Bonomi, che ne era il maggiore azionista, non sapeva più che pesci pigliare... Del resto, avrete sicuramente notato, in qualche servizio televisivo, che quasi sempre gli agenti di borsa comunicano tra loro grazie a strani segni, spesso molto buffi e risibili.

Assegnare un nome di fantasia ai titoli corrisponde, quindi, al comunicare al computer, mediante "segni" strani, la vostra intenzione di acquistare o vendere detrminati titoli senza che gli altri capiscano i vostri movimenti.

Dunque, tirando le somme, è bene fare le operazioni in modo avveduto e cercare di scoprire le proprie carte il meno possibile, affinchè gli avversari non abbiano a trarre vantaggi dai nostri movimenti.

Ovviamente sarà compito vostro ricordare che:

PIPPO = FIAT

ENEL = COMMODORE

....

oppure trascrivere il tutto su un foglio di carta (da custodire gelosamente...).

Analogamente il compito degli avversari consisterà anche nello scoprire il codice impostato dagli altri, in modo da adeguarsi con intelligenza alle varie operazioni.

Terminata questa fase, comincia il gioco vero e proprio.

La prima schermata (secondo riquadro di figura 1) presenta il listino con i titoli, le due quotazioni del giorno precedente e l'odierna. Eventuali differenze tra le due cifre vanno studiate attentamente, perchè soltanto da queste (oltre

che dall'attività... spionistica) potete avere utili informazioni e cercare di capire quanto e cosa hanno in mano i vostri amici-nemici.

Forse è scontato, ma vale la pena precisare che massicci acquisti di azioni ne faranno lievitare il valore. Nel caso opposto, il meccanismo è identico: vendere equivale a deprezzare il titolo.

Quando comprate (o vendete), il prezzo pagato non sarà quello che compare sul listino, ma una media che tiene conto della domanda-offerta.

Ecco perchè è necessario essere almeno in tre a giocare. Supponiamo, infatti, di essere soltanto in due e che, in una fase di compravendita, decidiamo di acquistare 500 azioni della società Fiat che, per noi, è codificata in PIPPO. Poichè l'avversario ha diritto, in questa fase del gioco, a controllare ciò che digitiamo, sarà sufficiente, per lui, non trattare i titoli PIPPO in questa transazione. Al termine della fase basterà rintracciare sul tabellone il titolo che ha incrementato il proprio valore per individuare il codice da noi assegnato a PIPPO.

Se si è in tre, invece, non sarà così facile scoprirlo perchè le quotazioni del tabellono sono influenzate da due squadre. E' come risolvere una sola equazione con due incognite: le soluzioni sono infinite.

L'andamento delle quotazioni, nel caso del livello di difficoltà 2, oltre a essere influenzato dalla richiesta, subisce anche il cosiddetto "umore" del mercato.

Nel programma sono inserite alcune variabili casuali che fanno oscillare, per periodi più o meno lunghi, i valori di ciascun titolo.

Queste sono da considerare come fattori imponderabili, dipendenti dalla somma e spesso dalla sinergia di tutti gli aspetti del mondo economico internazionale. Nella realtà i cosiddetti fattori imprevisti possono essere della natura più varia: crisi politiche, scioperi dei minatori, restrizioni economiche...

Figura 3 - Calcolo delle quotazioni. La quotazione corrente è calcolata in base alla quotazione precedente e al numero delle azioni acquistate o vendute (movimenti). Quest'ultimo valore influisce su un indice che sarà rispettivamente usato come fattore, se le azioni comprate sono maggiori di quelle vendute (il prezzo sale), oppure come divisore nel caso contrario (il prezzo scende).
Infine fattori casuali interverranno oppure no in relazione al livello di gioco scelto.

```
100 REM  BORSA
110 :
120 REM  GIOCO/SIMULAZIONE
130 REM  PER QUALSIASI COMPUTER
140 :
150 REM  DI FLAVIO MOLINARI
160 :
170 REM ***********************
    **
180 REM *       INIZIO
    *
190 REM ***********************
    **
200 NT=8:REM  N.TITOLI
210 NS=3:REM  N. SQUADRE
220 CP=100000000:REM  CAPITALE
    INIZIALE
230 AP=5000:REM  NUMERO AZIONI
    INIZIALI
240 M2=.008:REM  INDICE ANDAMEN
    TO DEL MERCATO
250 M5=1:REM  DEPREZZAMENTO
260 :
270 DIM LS$(20,3):REM  TITOLI
280 DIM LS(20,4):REM  LISTINO A
    ZIONI
290 DIM LZ(20,3):REM  MOVIMENTI
300 DIM Q(20,3):REM  POSIZIONI
    TITOLI
310 DIM QZ(20,1):REM  QUOTAZION
    I
320 DIM R(20):REM  SWITCH DI IN
    IZIO OSCILLAZIONI
330 DIM G(20):REM  GIORNI DI OS
    CILLAZIONE
340 :
350 REM  CAPITALE INIZIALE
360 LS(20,1)=CP
370 LS(20,2)=CP
380 LS(20,3)=CP
390 FOR Q=1 TO NT
400 READ LS$(Q,0):REM  TITOLO
410 READ QZ(Q,0):QZ(Q,1)=QZ(Q,0
    ):REM  QUOTAZIONE INIZIALE
420 LS(Q,1)=AP:LS(Q,2)=AP:LS(Q,
    3)=AP
430 NEXT
440 :
450 DATA FIAT,4530
460 DATA COMMODORE,6700
470 DATA ORO,19900
480 DATA SYSTEMS,5200
490 DATA GEMINA,1550
500 DATA BUITONI,4020
510 DATA PIRELLI,3200
520 DATA DOLLARI,1850
530 :
540 REM ***********************
    **
550 REM *   CODIFICA INIZIALE
    *
560 REM *        DEI TITOLI
    *
570 REM ***********************
    **
580 PRINTCHR$(147)SPC(250):INPU
    T "LIVELLO (1/2)";LV
590 IF LV<1 OR LV>2 THEN 580
600 FOR Q1=1 TO NS
610 PRINTCHR$(147):REM  CLEAR
620 PRINT"CODIFICA DEI TITOLI S
    QUADRA"Q1
630 PRINT
640 FOR Q2=1 TO NT
650 PRINTLS$(Q2,0);:Q$=""
660 INPUT Q$:IF Q$="" THEN 650
670 LS$(Q2,Q1)=Q$
680 NEXT
690 :
700 REM  DISTRIBUZIONE CASUALE
    TITOLI
710 FOR Q=1 TO NT:Q(Q,Q1)=0:NEX
    T
720 FOR Q=1 TO NT
730 RN=INT(RND(1)*NT)+1
740 IF Q(RN,Q1)<>0 THEN 730
750 Q(RN,Q1)=Q:NEXT
760 PRINT:PRINT
770 GOSUB 2220:REM  PREMI <RETU
    RN>
780 NEXT
790 :
800 REM ***********************
    **
810 REM *    COMPRO/VENDO
    *
820 REM ***********************
    **
```

```
830 M1=LS(20,1)+LS(20,2)+LS(20,
    3)+LS(19,1)+LS(19,2)+LS(19,
    3)
840 GOSUB 1540:REM  QUOTAZIONI
850 IF LV=2 THEN GOSUB 1990:REM
     MERCATO
860 GOSUB 1800:REM  LISTINO
870 GOSUB 2230:REM  <RETURN>
880 REM  AZZERA MOVIMENTI
890 FOR Q=1 TO NT
900 LS(Q,4)=0:NEXT
910 :
920 FOR Q1=1 TO NS
930 FOR Q=1 TO NT:LZ(Q,Q1)=0:NE
    XT
940 CT=0:LT=0
950 CT=CT+1:IF CT>4 THEN 1440
960 PRINTCHR$(147)"AZIONI SQUAD
    RA"Q1:PRINT
970 FOR Q=1 TO NT:Z=Q(Q,Q1)
980 PRINTLS$(Z,Q1) TAB(10)LS(Z,
    Q1)
990 NEXT
1000 PRINT
1010 PRINT"C) COMPRO"
1020 PRINT"V) VENDO"
1030 PRINT"F) FINE"
1040 PRINT
1050 GET Q$:IF Q$="" THEN 1050
1060 IF Q$="C" THEN 1140
1070 IF Q$="V" THEN 1310
1080 IF Q$="F" THEN 1440
1090 GOTO 1050
1100 :
1110 REM ***********************
     **
1120 REM *       COMPRO
      *
1130 REM ***********************
     **
1140 INPUT "COSA COMPRI";LS$
1150 S1=0:FOR Q=1 TO NT
1160 IF LS$(Q,Q1)=LS$ THEN S1=Q
1170 NEXT
1180 IF S1=0 THEN PRINT"TITOLO I
     NESISTENTE":GOSUB 2230:GOTO
      950
1190 PRINT
1200 INPUT "QUANTITA'";QU
1210 SP=QZ(S1,1)*QU:REM  SPESA
1220 IF LT+SP>LS(20,Q1) THEN PRI
     NT"NON HAI ABBASTANZA SOLDI
      !":GOSUB 2230:GOTO 950
1230 LS(S1,Q1)=LS(S1,Q1)+QU:REM
      AGGIORNA LISTINO
1240 LZ(S1,Q1)=LZ(S1,Q1)+QU:LT=L
     T+SP
1250 LS(S1,4)=LS(S1,4)-QU:REM  M
     OVIMENTI
1260 GOTO 950
1270 :
1280 REM ***********************
     **
1290 REM *       VENDO
      *
1300 REM ***********************
     **
1310 INPUT "COSA VENDI";LS$
1320 S1=0:FOR Q=1 TO NT
1330 IF LS$(Q,Q1)=LS$ THEN S1=Q
1340 NEXT
1350 IF S1=0 THEN PRINT"TITOLO I
     NESISTENTE":GOSUB 2230:GOTO
      950
1360 PRINT
1370 INPUT "QUANTITA'";QU
1380 SP=QZ(S1,1)*QU
1390 IF LS(S1,Q1)<QU THEN PRINT"
     NON DISPONIBILE!":GOSUB 223
     0:GOTO 950
1400 LS(S1,Q1)=LS(S1,Q1)-QU
1410 LZ(S1,Q1)=LZ(S1,Q1)-QU:LT=L
     T-SP
1420 LS(S1,4)=LS(S1,4)+QU
1430 GOTO 950
1440 GOSUB 2220
1450 AZ=0:FOR Q=1 TO NT
1460 AZ=AZ+LS(Q,Q1)*QZ(Q,1):NEXT
1470 LS(19,Q1)=AZ:NEXT
1480 GOTO 830
1490 :
1500 REM ***********************
     **
1510 REM *  AGGIORNA QUOTAZIONI
      *
1520 REM ***********************
     **
1530 REM  QUOTAZIONE DEL GIORNO
```

```
     PRECEDENTE
1540 FOR Q=1 TO NT
1550 QZ(Q,0)=QZ(Q,1):NEXT
1560 :
1570 FOR Q=1 TO NT
1580 MV=LS(Q,4):REM  MOVIMENTI
1590 IN=ABS(MV)/(AP*NS)
1600 IF MV<0 THEN QZ(Q,1)=INT(QZ
     (Q,1)*(1+IN))
1610 IF MV>0 THEN QZ(Q,1)=INT(QZ
     (Q,1)/(1+IN))
1620 IF MV=0 THEN QZ(Q,1)=INT(QZ
     (Q,1)*M5)
1630 IF ABS(MV)>30 THEN QZ(Q,1)=
     INT(QZ(Q,1)*(2-M5))
1640 NEXT
1650 :
1660 REM  DISPONIBILE IN AZIONI
1670 FOR Q1=1 TO NS:AZ=0:AS=0
1680 FOR Q=1 TO NT
1690 AZ=AZ+LS(Q,Q1)*QZ(Q,1)
1700 AS=AS+LZ(Q,Q1)*(QZ(Q,1)+QZ(
     Q,0))/2
1710 NEXT
1720 LS(19,Q1)=AZ
1730 LS(20,Q1)=LS(20,Q1)-AS
1740 NEXT
1750 RETURN
1760 :
1770 REM ***********************
     **
1780 REM *    STAMPA LISTINO
      *
1790 REM ***********************
     **
1800 PRINTCHR$(147)
1810 PRINT"TITOLO" TAB(10)"QUOTA
     Z QUOTAZ"
1820 PRINT TAB(10)"PRECED ODIERN
     A"
1830 PRINT
1840 FOR Q=1 TO NT
1850 PRINTLS$(Q,0) TAB(9)QZ(Q,0)
      TAB(16)QZ(Q,1)
1860 NEXT
1870 PRINT:PRINT TAB(7)"LIQUIDO"
      TAB(18)"AZIONI" TAB(29)"TO
     TALE"
1880 FOR Q=1 TO NS
1890 PRINT"SQ"Q; TAB(5)LS(20,Q);
      TAB(16)LS(19,Q);
1900 PRINT TAB(27)LS(20,Q)+LS(19
     ,Q)
1910 NEXT
1920 PRINT
1930 RETURN
1940 :
1950 REM ***********************
     **
1960 REM * MOVIMENTI DEL MERCATO
      *
1970 REM *     PSEUDOCASUALI
      *
1980 REM ***********************
     **
1990 FOR Q=1 TO NT
2000 IF R(Q)=0 THEN GOSUB 2050:R
     (Q)=S:G(Q)=G
2010 IF R(Q)=1 THEN GOSUB 2110
2020 IF R(Q)=2 THEN GOSUB 2150
2030 NEXT:RETURN
2040 :
2050 RN=RND(1):S=0
2060 IF RN<.1 THEN S=1
2070 IF RN<.2 AND RN>.1 THEN S=2
2080 G=INT(RND(1)*10)+2:REM  N.
     GIORNI DI VARIAZIONE
2090 RETURN
2100 :
2110 QZ(Q,1)=INT(QZ(Q,1)*(1-M2))
2120 G(Q)=G(Q)-1:IF G(Q)=0 THEN
     R(Q)=0
2130 RETURN
2140 :
2150 QZ(Q,1)=INT(QZ(Q,1)*(1+M2))
2160 G(Q)=G(Q)-1:IF G(Q)=0 THEN
     R(Q)=0
2170 RETURN
2180 :
2190 :
2200 :
2210 REM  *** PREMI <RETURN> ***
2220 PRINT"FINE SQUADRA"Q1
2230 PRINT"PREMI <RETURN> PER CO
     NTINUARE"
2240 GET Q$:IF Q$<>CHR$(13) THEN
      2240
2250 RETURN
```

computer service (VENDITA PER CORRISPONDENZA)

ACCESSORI PER COMPUTER COMMODORE

GRUPPO CONTINUITÀ
Fornito senza le 12 batterie a stilo ricaricabili. Consente il funzionamento del Vostro computer Commodore C64 o VIC 20 in assenza di corrente. Durata di funzionamento 30 minuti. Ricarica tramite alimentatore Commodore.

KIT ALLINEAMENTO TESTINA
Composto dal cacciavite, nastro di controllo e strumento di taratura con monitor audio permette il perfetto allineamento dei registratori digitali anche con nastri commerciali.

VELOCIZZATORE DI CARICAMENTO FLOPPY
Cartridge con un insieme di utility residenti su ros per velocizzare il drive nel Commodore 64.

INTERFACCIA RADIO
Indispensabile per registrare con registratore Commodore modello "C2N" i programmi speciali per computer trasmessi dalle emittenti radio.

CUFFIA PER COMMODORE C 64
Leggerissima permette l'ascolto personale del computer evitando di disturbare durante i giochi.

COPIATORE PROGRAMMI
Dispositivo hardware per effettuare copie di nastri protetti o turbo utilizzando due registratori Commodore o compatibili.

DUPLICATORE CASSETTE
Indispensabile per realizzare delle copie, con un registratore normale, di un nastro protetto o con caricamento turbo

Articolo	Art.	L.
Bus quadrislot	Art. CD 100	L. 55.000
Interfaccia cassette	Art. CD 101	L. 30.000
Duplicatore cassette	**Art. CD 102**	**L. 30.000**
Copiatore programmi	**Art. CD 103**	**L. 30.000**
Interfaccia radio	**Art. CD 104**	**L. 30.000**
Kit allineamento testina	**Art. CD 105**	**L. 47.000**
Alimentatore per C64 e VIC 20	Art. CD 106	L. 45.000
Gruppo continuità (fornito senza le 12 batterie a stilo ricaricabili)	**Art. CD 107**	**L. 66.000**
Pacco batterie (12 stilo 1,2 Volt ricaricabili	Art. CD 117	L. 52.000
Commutatore antenna TV/computer	Art. CD 108	L. 9.500
Tasto reset	Art. CD 109	L. 5.500
Interfaccia Centronics	Art. CD 112	L. 104.000
Espansione di memoria per C 16	Art. CD 114	L. 158.000
Velocizzatore di caricamento flop.	**Art. CD 115**	**L. 49.000**
Espansione di memoria per VIC 20 16K	Art. CD 116	L. 112.000
Modulatore Executive	Art. CD 120	L. 72.000
Penna ottica grafica	Art. CD 121	L. 45.000
Tavoletta grafica	Art. CD 130	L. 238.000
Multipresa con filtro - 2 prese	Art. CD 140	L. 41.000
Cuffia per Commodore C 64	**Art. CD 150**	**L. 19.000**
Stabilizzatore elettronico di tensione 500 W	Art. CD 160	L. 430.000
Gruppo di cintinuità 60 W	Art. CD 170	L. 400.000
Gruppo di continuità 200 V	Art. CD 180	L. 802.000
Inverter 12 Volt cc. 220 Volt ca. 100 Watt	Art. CD 190	L. 297.000
Cavo alimentazione	Art. CD 200	L. 4.600
Cavo drive o stampante Commodore	Art. CD 205	L. 8.500
Prolunga per Joystick - mt. 3	Art. CD 210	L. 25.000
Prolunga per cavo TV - mt. 3	Art. CD 215	L. 12.500
Cavo audio - mt. 6	Art. CD 220	L. 15.500
Adattatore Joistick (Atari e C64 al C 16)	Art. CD 225	L. 10.500
Adattatore registratore per C 16	Art. CD 226	L. 19.500
Mascherina antiriflesso 12"	Art. CD 300	L. 35.000
Nastro inchiostrato per Tally - mt. 80	Art. CD 610	L. 16.500
Nastro inchiostrato per Tally - mt. 180	Art. CD 611	L. 16.500
Nastro inchiostrato per Tally 1000 e Honeywell	Art. CD 612	L. 9.500
Nastro inchiostrato per Commodore MRS 801	Art. CD 614	L. 13.000
Nastro inchiostrato per Commodore MPS 802	Art. CD 616	L. 18.000
Nastro inchiostrato per Commodore MPS 803	Art. CD 618	L. 19.500
Mause per Commodore C 64	Art. CD 860	L. 240.000
Pacco carta lettura facilitata 24" x 11" modulo da 500 fogli con bordi a strappo	Art. CD 630	L. 13.500
Supporto stampante porta carta in plexiglass "fume" - normale	Art. CD 660	L. 59.000
Supporto stampante porta carta in plexiglass "fume" - rinforzato	Art. CD 670	L. 80.000
Floppy disk 5" singola faccia doppia densità "ODP" - conf. 10 pezzi	Art. CD 700	L. 40.000
Floppy disk 5" singola faccia doppia densità "CBS" - conf. 10 pezzi	Art. CD 702	L. 38.000
Floppy disk 5" singola faccia doppia densità "VERBATIM" - conf. 10 pezzi	Art. CD 704	L. 42.000
Floppy disk 5" singola faccia doppia densità "DYSAN" - conf. 10 pezzi	Art. CD 706	L. 68.000
Nastri magnetici C 10 digitali - conf. 10 pezzi	Art. CD 712	L. 20.000
Nastri magnetici C 15 digitali	Art. CD 714	L. 21.000
Copritastiera in plexiglass per C64 - C16 e VIC 20	Art. CD 750	L. 16.000
Copritastiera in stoffa per C64 - C16 e VIC 20	Art. CD 760	L. 10.500
Vaschetta portafloppy in plexiglass per 40 dischi con chiave	Art. CD 770	L. 30.000
Vaschetta portafloppy in plexiglass per 90 dischi con chiave	Art. CD 780	L. 37.000
Kit pulizia testine registratore	Art. CD 815	L. 13.500
Kit pulizia disk drive	Art. CD 820	L. 26.000
Kit pulizia tastiera	Art. CD 830	L. 16.500
Foratore disk in plastica (per utilizzare la seconda faccia dei dischi)	Art. CD 840	L. 10.000
Foratore disk in metallo "tako"	Art. CD 849	L. 14.000
Joystick Spectravideo II	Art. CD 850	L. 27.000
Joystick a Microswitch	Art. CD 851	L. 52.500
Joystick senza fili con unità ricevente (funziona a batteria)	Art. CD 852	L. 98.000
Joystick per Commodore 16 (originale)	Art. CD 130	L. 29.500

TUTTI I PREZZI SONO COMPRENSIVI DI IVA
NON SI ACCETTANO ORDINI INFERIORI A L. 30.000
CONTRIBUTO FISSO SPESE DI SPEDIZIONE L. 5000

SI ACCETTANO ANCHE ORDINI TELEFONICI AI NUMERI 0522/661647-661471

BUONO DI ORDINAZIONE

NOME - COGNOME

INDIRIZZO   N.

C.A.P.   CITTA   PROVINCIA

VOGLIATE INVIARMI IN CONTRASSEGNO

N.	Art.	L.
N.	Art.	L.
N.	Art.	L.
SPESE SPEDIZIONE		L. 5.000
PAGHERÒ AL POSTINO		L.

COMPUTER SERVICE VIA A. MANZONI, 49 - 42017 NOVELLARA (RE) - TEL. (0522) 661647

Enciclopedia di routine

a cura di Alessandro de Simone

12300 MCD e mcm

(Per qualsiasi Commodore)

Programmi sui numeri primi, risoluzione di equazioni o ricerca di divisori sono già stati proposti su queste stesse pagine e probabilmente i più assidui lettori di Commodore Computer Club riconosceranno nella routine lo stesso procedimento adottato in un listato di qualche numero addietro. Ve lo riproponiamo in una nuova veste, appunto come routine dell'Enciclopedia, anche per dare l'opportunità ai nuovi lettori di usufruire di questo interessante programma o (perchè no?!), di cogliere l'occasione per fare un po' di "didattica" in maniera non troppo seriosa.

La ricerca dei divisori comuni a due numeri (MCD) è spesso richiesta nella risoluzione di problemi matematici: il metodo più intuitivo, ma anche il più noioso, consiste nello scomporre entrambi i valori in fattori primi e poi moltiplicare fra loro i divisori che compaioni da ambo le parti. Esiste però un'altra via, detta delle divisioni successive, molto più elegante e, soprattutto, più rapida, specialmente se i numeri in questione sono piuttosto grandi.

Ora, senza annoiarvi troppo con spiegazioni di tipo scolastico, vediamo in un esempio pratico come funziona l'algoritmo usato nel programma. Supponiamo di voler determinare il massimo comun divisore di 714 e 966:

966/714=1
resto: 252

Prendiamo il più piccolo dei due valori (714) e il resto ottenuto e dividiamoli a loro volta:

714/252=2
resto: 210

E ancora:

252/210=1
resto: 42
210/42=5
resto: 0

Il numero cercato è 42, vale a dire il primo divisore che fornisce un resto nullo. Non chiedeteci la ragione; il sistema funziona perfettamente anche senza saperne il perchè!

Poniamo fina all'estemporanea lezione di algebra e veniamo alla nostra routine.

Le variabili Y1 e Y2 contengono i due numeri da elaborare, mentre al "ritorno" abbiamo:

XM=MCD

Con poca spesa possiamo calcolare anche il minimo comune multiplo:

YM=mcm

Ricordiamo che il mcm si ricava da:

mcm=Y1*Y2/MCD

Flavio Molinari

```
100 REM  ESEMPIO D'USO
110 REM  MASSIMO COMUN DIVISORE
      E
115 REM  MINIMO COMUNE MULTIPLO
120 REM  METODO DELLE DIVISIONI
      SUCCESSIVE
130 REM  QUALSIASI COMMODORE
140 :
150 Y1=0:INPUT "PRIMO NUMERO";Y
    1
160 Y2=0:INPUT "SECONDO NUMERO"
    ;Y2
165 GOSUB 12300
170 IF X0$="ERR" THEN PRINT"NUM
    ERI INTERI DIVERSI DA ZERO"
```

```
    :PRINT:GOTO 150
  180 PRINT
  190 PRINT"MASSIMO COMUN DIVISOR
      E=";X5
  200 PRINT"MINIMO COMUNE MULTIPL
      O=";Y0
  210 PRINT:PRINT:GOTO 150
  220 :
12300 X0$="":IF Y1=0 OR Y2=0 THEN
       X0$="ERR":RETURN:REM  NUME
      RI <> DA 0
12305 IF Y1<>INT(Y1) OR Y2<>INT(Y
      2) THEN X0$="ERR":RETURN:RE
      M  NUMERI INTERI
12310 X1=ABS(Y1):X2=ABS(Y2)
12320 X3=INT(X1/X2)
12330 X4=X1-X2*X3
12340 IF X4=0 THEN X5=X2:Y0=ABS(Y
      1*Y2/X2):RETURN
12350 IF X2/X4=INT(X4/X2) THEN X5
      =X4:Y0=ABS(Y1*Y2/X4)
12360 X1=X2:X2=X4:GOTO 12320
12365 REM  VARIABILI DI LAVORO: X
      3,X4,X0$
12370 REM  INPUT: Y1,Y2
12375 REM  OUTPUT: X5=MCD, Y0=MCM
12399 REM  M.C.D. E M.C.M.
```

12400 Print using

(Per qualsiasi Commodore)

E' un altro problema di trattamento di numeri, ma stavolta la questione è squisitamente informatica. con questa routine, infatti, cercheremo di sopperire, almeno in parte, alla mancanza dell'istruzione PRINT USING nell'interprete Basic del Commodore 64 e Vic 20.

Il listato gira in effetti su qualsiasi computer Commodore, ma se possedete un C-16, un Plus/4 oppure un C-128, non vi sarà di grande utilità, dato che l'istruzione che cercheremo di simulare esiste già all'interno del Basic 3.5 o del più potente Basic 7.0.

Ai numerosi utenti del C-64 si sarà presentata, almeno una volta, la necessità di visualizzare numeri in maniera ordinata e leggibile e di dover ricorrere, per raggiungere lo scopo, ad artefici che spesso non davano risultati

soddisfacenti.
Infatti se scriviamo:

10 PRINT 23
20 PRINT 123.12
30 PRINT 9999

dando il RUN avremo:

23
123.12
9999

Nel caso di programmi che prevedono l'output su video o su stampante, numero così disposti farebbero storcere il naso anche alla persona meno esigente in fatto di preziosismi estetici.

Con la routine che segue potremmo definire il tipo dei dati (intero o decimale), le dimensioni dei campi e stampare il numero assegnandolo alla variabile Y0. Ad X0$ dovrà corrispondere una stringa del tipo:

X0$="   ,   "
in cui il numero di caratteri "griglia" (#) posti a sinistra e a destra del punto definiscono l'ampiezza del campo numerico intero e decimale. Se il numero di questi ultimi è maggiore di quello definito nel campo, il valore sarà arrotondato:

Se Y0 vale....	e X0$ è.....	Y0$ risulta:
123.456	" # "	"123.46"
78.9	" # "	"78.90"
1234	" # "	" 1234"

Se il campo della parte intera è insufficiente a contenere il valore, viene stampato il simbolo della percentuale (%) in testa al numero, per avvertire che c'è un errore. Esempio:

Y0=123.5 X0$="   #   " Y0$="%123.50"

L'ultima opzione è il completamento a sinistra con zeri (una nota personale che si discosta un po' dal Basic microsoft), ottenibile ponendo uno 0 in testa alla variabile X0$:

Y0=3 X0$="0   #   " Y0$="0003"

Flavio Molinari

```
100 REM  ESEMPIO D'USO
110 REM  PRINT USING
120 REM  FORMATTAZIONE DATI NUM
    ERICI
130 REM  QUALSIASI COMPUTER
140 :
150 PRINT"ESEMPIO 1: STAMPA DI
    NUMERI INTERI E CON DECIMAL
    I"
160 X0$="###.##"
170 REM  3 CIFRE PER LA PARTE N
    TERA E 2 PER LA PARTE DECIM
    ALE
180 Y0=23.456:GOSUB 12400:PRINT
    Y0$
190 Y0=567    :GOSUB 12400:PRINT
    Y0$
200 Y0=99.9   :GOSUB 12400:PRINT
    Y0$
201 Y0=.9     :GOSUB 12400:PRINT
    Y0$
225 :
230 PRINT"ESEMPIO 2: STAMPA CON
     COMPLETAMENTO DI ZERI A SI
    NISTRA"
240 X0$="0###"
250 Y0=2423   :GOSUB 12400:PRINT
    Y0$
260 Y0=323.22:GOSUB 12400:PRINT
    Y0$
270 Y0=22     :GOSUB 12400:PRINT
    Y0$
290 :
9999 END
12400 X9=0:X8=0:XW=0:XE=1:XD=1
12405 XS$="         ":FOR XQ=1 TO
     LEN(X0$):XX$=MID$(X0$,XQ,1
    ):REM  9 SPAZI
12410 IF XX$<>"." AND XX$<>"#" AN
    D XX$<>"0" THEN X0$="ERR":R
    ETURN
12412 IF XX$="0" THEN XS$="000000
    000"
12415 IF XX$<>"." AND XW=0 THEN X
    8=X8+1:XD=XD*10
12420 IF XX$="." THEN XW=1:GOTO 1
    2430
12425 IF XW=1 THEN X9=X9+1:XE=XE*
    10
12430 NEXT
12435 Y0=INT(Y0*XE+.5)/XE:Y0$=STR
    $(Y0)
12440 Y0$=MID$(Y0$,2,LEN(Y0$)-1)
12445 IF Y0=INT(Y0) THEN Y0$=Y0$+
    ".0"
12455 XZ=LEN(STR$(INT(Y0))):Y1$=L
    EFT$(Y0$,XZ-1):Y2$=MID$(Y0$
```

```
12540 RETURN
12555 REM  X0$,XV,XN,X3,X2,X1,X1$
12560 REM  XV=VECCHIO COLORE
12565 REM  XN=NUOVO COLORE
12599 REM  CAMBIA COLORI
```

12600 Text copy

(Per C-64, C-16 e stampante MPS 803 o compatibili)

Un'utility che non può mancare ai possessori della piccola stampante MPS 803.

Tutto ciò che appare sul video al momento della "chiamata" del sottoprogramma verrà copiato fedelmente su carta nel modo normale se la stringa X0$ contiene i caratteri "NOR", oppure in "allargato" (se X0$="ALL").

Solo un piccolo neo: per ragioni tecniche il carattere di virgolette (") è sostituito dal carattere apostrofo (shift e 7) per evitare problemi nella stampa di eventuali simboli grafici in reverse.

Approfittiamo dell'occasione per vedere alcuni codici comando utilizzati nel programma e che per la MPS 803 corrispondono ad altrettanti modi di stampa:

CHR$(15): stampa con carattere normale;
CHR$(145): seleziona il set maiuscolo/grafico.

I due modi appena visti sono impostati automaticamente al momento dell'accensione della stampante. Volendo è possibile cambiare il tipo di stampa con:

CHR$(14): stampa allargata (doppia rispetto al normale);
CHR$(17): seleziona il set maiuscolo/minuscolo.

Infine due codici che, per la stampante, hanno le medesime funzioni che sul video:

CHR$(18): stampa in reverse;
CHR$(146): disabilita il modo reverse.

La routine è facilmente adattabile al C-16. In tal caso sostituire XV=1024 con XV=3072 (inizio memoria video). Inoltre nel C-64 la locazione 53272 contiene l'indicazione dell'attuale set di caratteri utilizzato, maiuscolo/grafico o maiuscolo/minuscolo: sostituitela con l'indirizzo della locazione che nel C-16 espleta le medesime funzioni, oppure lasciate le cose così come sono, dato che il programma funzionerà egualmente, con l'unica limitazione che la stampante utilizzerà solamente il set di caratteri impostato in precedenza.

Flavio Molinari

```
100 REM  ESEMPIO D'USO
110 REM  TEXT COPY
120 REM  PER C64 O C16
130 REM  STAMPANTE MPS 803
132 :
134 REM  PER C16: SOSTITUIRE XV
    =1024 CON XV=3072
140 :
150 X0$="NOR":REM  CARATTERE NO
    RMALE
160 REM  X0$="ALL" PER CARATTER
    E ALLARGATO
170 GOSUB 12600
180 END
190 :
12600 XN=0:CLOSE 4:OPEN 4,4
12605 IF X0$="ALL" THEN PRINT#4,C
      HR$(14)
12610 IF X0$="NOR" THEN PRINT#4,C
      HR$(15)
12615 XV=1024:REM  INIZIO MEMORIA
       VIDEO
12620 IF PEEK(53272)=21 THEN X2$=
      CHR$(145):REM  SET MAIUSCOL
      O/GRAFICO
12625 IF PEEK(53272)=23 THEN X2$=
      CHR$(17):REM  SET MAIUSCOLO
      /MINUSCOLO
12630 FOR XX=0 TO 999:X1=PEEK(XX+
      XV)
12635 IF X1=34 THEN X1=39
12640 IF X1=32 OR X1=160 THEN Y1=
      Y2:GOTO 12650
12645 Y1=(X1 AND 127) OR ((X1 AND
       64)*2) OR ((64-X1 AND 32)*
      2)
12650 Y1$=CHR$(Y1)
12655 IF X1>127 THEN Y1$=CHR$(18)
      +Y1$+CHR$(146)
12660 PRINT#4,Y1$;
12665 XN=XN+1:IF XN>39 THEN PRINT
      #4,X2$:XN=0
12670 NEXT:CLOSE 4:RETURN
12680 REM  XN,XV,X1$,X2$,Y1,X0$,Y
      1$
12699 REM  TEXT COPY MPS 803-COMP
      ATIBILI
```

```
       ,XZ)
12460 Y1$=RIGHT$(XS$+Y1$,X8)
12465 Y2$=LEFT$(Y2$+"000000000",X
      9+1)
12470 IF X9=0 THEN Y2$=""
12475 IF XD<INT(Y0) THEN Y0$="%"+
      MID$(Y0$,1,LEN(STR$(INT(VAL
      (Y0$))))-1)+Y2$:RETURN
12480 Y0$=Y1$+Y2$:RETURN
12499 REM  PRINT USING
```

12500 Cambia colori

(Per C-64)

Il sottoprogramma, che permette di cambiare rapidamente i colori della pagina testo, può essere impiegato in tutte le applicazioni ove siano richiesti effetti particolari (ad esempio il lampeggio di porzioni di schermo o il cambio di uno sfondo), non ottenibili con i normali comandi Basic disponibili sul Commodore 64. Infatti ci si avvale di una breve subroutine in linguaggio macchina, allocata nella parte bassa della memoria (a partire da 828), che lavora sulla memoria colore (55296-56295).

Ponendo ad esempio:

XV=0 (vecchio colore: nero)
XN=8 (nuovo colore: rosso)

tutti i caratteri sullo schermo di color nero, sempre che ne esistano, saranno cambiati istantaneamente (o quasi) in rosso.

L'allocazione della routine avviene nel consueto modo, "inventato" per l'occasione per l'Enciclopedia di routine: i codici in LM sono memorizzati nella variabile stringa X1$ e letti per mezzo della funzione MID$.

Chi ha seguito la rubrica sin dalle prime puntate probabilmente ne conoscerà la ragione, ma di tanto in tanto vale la pena ricordare che si è optato per questa soluzione, per non creare problemi nel caso la routine venga utilizzata all'interno di programmi che già contengono istruzioni READ a DATA.

Le linee contenenti la parola "CANCELLARE" effettuano un controllo sull'esattezza dei valori della parte di programma in LM, dato che una sola cifra digitata in modo errato porterebbe molto probabilmente all'inchiodamento del computer. Dopo aver fatto girare una volta il programma e averne verificato il corretto funzionamento, potete quindi cancellare le righe contenenti tale controllo.

Flavio Molinari

```
  100 REM  ESEMPIO D'USO
  110 REM  CAMBIA COLORI
  120 REM  SOLO COMMODORE 64
  130 :
  140 PRINTCHR$(147):REM  CLEAR S
      CREEN
  150 XX$=" PROVA ROUTINE CAMBIA
      COLORI"
  155 PRINTCHR$(158):REM  GIALLO
  160 FOR XX=1 TO 10:PRINTXX$;:NE
      XT
  170 PRINTCHR$(144):REM  NERO
  180 FOR XX=1 TO 10:PRINTXX$;:NE
      XT
  210 XV=0:XN=5:REM  NERO IN VERD
      E
  220 GOSUB 12500
  230 FOR XX=1 TO 500:NEXT
  240 XV=7:XN=0:REM  GIALLO IN NE
      RO
  260 GOSUB 12500
  270 FOR XX=1 TO 500:NEXT
  290 XV=5:XN=7:REM  VERDE IN GIA
      LLO
  300 GOSUB 12500
  310 FOR XX=1 TO 500:NEXT
  320 GOTO 210
  330 :
12500 X1$="169000133251169219"
12505 X1$=X1$+"133252162004160232
      "
12506 X1$=X1$+"136177251041015201
      "
12508 X1$=X1$+"000208004169000145
      "
12510 X1$=X1$+"251192000208239198
      "
12512 X1$=X1$+"252202208234096"
12515 X3=0:FOR X1=1 TO LEN(X1$) S
      TEP 3:X3=X3+VAL(MID$(X1$,X1
      ,3)):NEXT:REM  CANCELLARE
12520 IF X3<>5311 THEN PRINT"ERRO
      RE NEI DATA":END :REM  CANC
      ELLARE
12525 X2=0:FOR X1=1 TO LEN(X1$) S
      TEP 3:POKE 828+X2,VAL(MID$(
      X1$,X1,3)):X2=X2+1:NEXT
12527 IF XV<0 OR XV>255 OR XN<0 O
      R XN>255 THEN Y0$="ER":RETU
      RN
12530 POKE 846,XV:POKE 850,XN:SYS
      828
```

12700 Fill memoria

(Per C-64)

Tre esempi di utilizzo per un'utility dall'impiego universale. Il suo funzionamento è molto semplice: la parte realizzata in LM svolge le medesime funzioni del seguente micro programma Basic (nel caso, ad esempio, della memoria video):

```
90 INPUT X0
100 FOR Q=0 TO 999
110 POKE 1024+Q,X0
120 NEXT
```

Come avrete intuito, la routine proposta serve a "riempire" una zona di memoria RAM con un determinato valore (compreso tra 0 e 255), a una velocità ovviamente di gran lunga superiore a quella tipica del Basic.

Le opzioni sono quattro, selezionabili assegnando alla variabile stringa X0$ altrettanti nomi-codice:

con X0$="VIDEO" lo schermo sarà riempito con il carattere che corrisponde al codice schermo X0 (attenzione, è diverso dal codice ASCII, ad esempio la lettera A risponde al codice 1, mentre in ASCII vale 65: a riguardo consultate il manuale del computer dalla pagina 132);
con X0$="COLOR" tutti i caratteri presenti sullo schermo vengono "colorati" con il colore corrispondente a X0 (0=nero, 1=bianco, eccetera);
con X0$="GRAF1" imposta la pagina grafica del primo banco di memoria (8192-16384) al valore X0;
con X0$="GRAF2" come sopra per il secondo banco di memoria (24576-32764). Può tornare utile nel caso si disegni in alta risoluzione: ponendo X0=0 verrà "pulita" l'intera pagina grafica.

Flavio Molinari

```
100 REM  ESEMPIO D'USO
110 REM  FILL MEMORIA
120 REM  SOLO COMMODORE 64
130 :
140 REM  ESEMPIO 1: MEMORIA VID
    EO
150 X0$="VIDEO":X0=81:REM  CARA
    TTERE " COLLECT"
160 GOSUB 12700:FOR I=1 TO 1000
    :NEXT
170 :
180 REM  ESEMPIO 2: MEMORIA COL
    ORE
190 X0$="COLOR":X0=0:REM  NERO
200 GOSUB 12700:FOR I=1 TO 1000
    :NEXT
210 STOP
220 REM  ESEMPIO 3: HIRES BANCO
    1 (8192)
230 POKE 53272,PEEK(53272) OR 8
240 POKE 53265,PEEK(53265) OR 3
    2
260 X0$="GRAF1":X0=0:REM  PULIS
    CE PAGINA GRAFICA
270 GOSUB 12700:FOR I=1 TO 1000
    :NEXT
280 :
9999 END
12700 X1$="160000166251165252"
12705 X1$=X1$+"145253200208249230
      "
12710 X1$=X1$+"254202208244096"
12715 X3=0:FOR X1=1 TO LEN(X1$) S
      TEP 3:X3=X3+VAL(MID$(X1$,X1
      ,3)):NEXT:REM  CANCELLARE
12720 IF X3<>3283 THEN PRINT"ERRO
      RE NEI DATA":END :REM  CANC
      ELLARE
12725 X2=0:FOR X1=1 TO LEN(X1$) S
      TEP 3:POKE 828+X2,VAL(MID$(
      X1$,X1,3)):X2=X2+1:NEXT
12727 IF X0<0 OR X0>255 THEN Y0$=
      "ER":RETURN
12730 IF X0$="VIDEO" THEN POKE 25
      1,4:POKE 254,4
12735 IF X0$="COLOR" THEN POKE 25
      1,4:POKE 254,216
12740 IF X0$="GRAF1" THEN POKE 25
      1,32:POKE 254,32
12745 IF X0$="GRAF2" THEN POKE 25
      1,32:POKE 254,96
12750 POKE 252,X0:POKE 253,0:SYS8
      28:RETURN
12755 REM  X0$,X0,X1,X2,X3,X1$
12760 REM  X0$:ZONA MEMORIA, X0:V
      ALORE
12799 REM  FILL MEMORIA
```

12800 Bordo in technicolor

(Per C-64)

L'idea non è completamente nuova, ma coloro che sono abituati a vedere il bordo dello schermo nel solito modo

Come realizzare l'enciclopedia e utilizzarla nei propri listati.

Ai lettori che hanno acquistato per la prima volta questo numero di Commodore Computer Club, illustriamo qui di seguito, in breve, i vantaggi derivanti dalla raccolta proposta. Questa, a pensarci bene, è la versione "superiore" della rubrica "1 RIGA" e potrebbe anche denominarsi... "Una schermata"!

Oltre che utili per costituire un'enciclopedia, i brevissimi sottoprogrammi pubblicati su ogni numero, sono anche validissimi strumenti di studio per coloro che desiderano approfondire le proprie conoscenze del Basic, esaminando, senza fatica, particolari routine o insolite tecniche di programmazione.

- Dato che può esser "chiamata" più di una volta nel corso di un programma, nessuna routine contiene istruzioni del tipo DATA oppure DIM, allo scopo di non creare confusione col listato principale.
- Nessuna routine può far riferimento ad altre routine dell'enciclopedia.
- Nessuna routine può contenere variabili "banali" (A, A$, eccetera), ma solo variabili poco usate (X1$, X8, Y0%, eccetera).
- Ogni routine deve apparire, **per intero,** sullo schermo del computer e consentire, proprio per questo motivo, di essere esaminata comodamente.
- Ogni routine deve esser numerata secondo uno standard che ha la particolarità di esser ricordato facilmente:

Righe	Contenuto
XXY00	Prima riga del sottoprogramma
XXY89	Ultima riga utile del sottoprogramma
XXY90 REM	Prima riga di spiegazioni
XXY99 REM	Nome della subroutine

in cui XX sono due valori variabili da 10 a 63; Y è un carattere numerico compreso tra 0 e 9.

Qualsiasi subroutine, in altre parole, inizia con un numero, di cinque caratteri, che termina **sempre** con "00". La stessa subroutine, d'altra parte, ha l'ultima riga numerata con "99". Digitando, ad esempio: LIST 10800-10899
si avrà la certezza di veder apparire sullo schermo, **per intero,** la routine il cui nome si trova nella riga 10899.

Prima di accedere alla routine, è necessario assegnare, alle variabili indicate con REM da riga XXY89 a XXY98, particolari valori per il suo corretto funzionamento. Al "ritorno" una o più variabili conterranno il risultato dell'elaborazione.

In questo modo, per esser più chiari, è possibile simulare alcuni comandi di versioni Basic avanzate oppure, addirittura, creare nuove e inedite istruzioni. Ad esempio, il comando: SOUND 1,800,500
che, nel C-16, riproduce un suono di tonalità 800 tramite la voce 1 per la durata 500, potrebbe venir riprodotta, in un'ipotetica subroutine per il Commodore 64, con: X1=1:X2=800:X3=500:GOSUB12400
nell'ipotesi, ovviamente, che la routine in oggetto sia allocata da riga 12400 a 12499.

I listati pubblicati "girano" su ogni computer, salvo dove indicato diversamente.

E' ovvio che nel caso del Vic-20, (che, come è noto, ha uno schermo di soli 506 caratteri), le subroutine "universali" funzionano correttamente, ma non possono apparire per intero in una sola schermata.

Per quanto riguarda la digitazione, si tenga presente che sulla rivista, per motivi di chiarezza, i comandi e le istruzioni Basic sono separati tra loro da spazi bianchi. Nel digitare le linee di programma, pertanto, è opportuno ignorarli altrimenti si rischia di non restare in una sola schermata. Se, per esempio, leggete:

```
12100 X1=34:  X2  = SQR(X3) + LOG(X1)
```

digitate nel modo seguente:

```
12100 X1=34:X2=SQR(X3)+LOG(X1)
```

senza, cioè, alcun carattere di separazione tra comandi ed istruzioni.

Collaborazione dei lettori

La collaborazione dei lettori è gradita, purchè si provveda a inviare **almeno** tre sottoprogrammi per volta, su nastro, disco oppure output di stampante. I listati di routine che non rispettano lo standard adottato non potranno esser presi in considerazione.

Tutti i lavori pubblicati verranno compensati con prodotti della Systems Editoriale (cassette di programmi, libri, abbonamenti, copie arretrate, eccetera).

(monocromatico e con al massimo la possibiltà di cambiarne il colore), facendo girare questo programma resteranno piacevolmente sorpresi.

Come tutti saprete, il video del Commodore 64 è suddiviso in due zone ben distinte: una centrale, su cui scriviamo o disegnamo, e una cornice esterna, sulla quale è possibile, con i comandi Basic, intervenire solamente sul colore (locazione 53280).

Grazie al LM e a qualche "trucchetto", possiamo sfruttare anche questa zona di schermo, non dico per scriverci sopra (bisognerebbe essere proprio dei maghi!), ma per ottenere insoliti effetti da inserire nei nostri programmi, per renderli più vivaci.

Assegnando alla variabile X0$ la stringa "COLOR", il bordo verrà suddiviso in numerose bande orizzontali, la cui altezza in dot (puntini elementari) è data dal valore assegnato in precedenza alla variabile X0.

Valori che esprimono potenze di due forniscono un'immagine stabile (a parte un leggero sfarfallio). In tutti gli altri casi l'effetto è diverso: fate girare il breve programma dimostrativo per rendervene conto.

Per tornare alla normalità assegnate a X0$ una qualsiasi stringa diversa da "COLOR".

Vediamo brevemente come è stato ottenuto questo simpatico effetto.

Il Commodore 64 dispone di particolari locazioni di memoria che consentono di controllare la posizione del cannone elettronico (per intenderci è quello che visualizza le immagini sullo schermo). Le locazioni 53265 e 53266 (in esadecimale $D011 e $D012) contengono il numero della riga in cui si trova il cannone in esame, informazione ottenibile con un'operazione di lettura: PEEK(53266) o, in linguaggio macchina, con LDA $D012.

Viceversa, scrivendo un valore (POKE 53266,X oppure STA $D012), non ne viene mutata la posizione, ma si mette in condizione il VIC II di programmare un interrupt: nella nostra applicazione ogni volta che il fascio di elettroni si è spostato di 0X pixel (nel programma demo X0t8) viene incrementata la locazione 53280 che, come visto in precedenza, cambia il colore al bordo dello schermo suddividendolo in tante fasce orizzontali di tonalità differenti.

Poichè non è possibile dedicare, nella presente rubrica, molte pagine alla descrizione di un solo programma, ricordiamo che sul numero 18 di Commodore Computer Club l'argomento viene trattato in maniera approfondita, compreso il disassemblato della routine in LM.

Flavio Molinari

```
100 REM  ESEMPIO D'USO
110 REM  BORDO IN TECHNICOLOR
120 REM  SOLO PER COMMODORE 64
  130 :
  150 X0$="COLOR":X0=8:GOSUB 1280
      0
  160 FOR XX=1 TO 1000:NEXT
  170 X0$="":GOSUB 12800
  190 :
  210 X0$="COLOR":X0=5:GOSUB 1280
      0
  220 FOR XX=1 TO  400:NEXT
  230 X0$="":GOSUB 12800
  240 :
  250 :
 9999 END
12800 X1$="169127141013220169"
12805 X1$=X1$+"094141020003169003
      "
12810 X1$=X1$+"141021003169129141
      "
12815 X1$=X1$+"013220169001141026
      "
12820 X1$=X1$+"208173017208041127
      "
12825 X1$=X1$+"141017208096173025
      "
12830 X1$=X1$+"208041001208003076
      "
12835 X1$=X1$+"188254141025208173
      "
12840 X1$=X1$+"133003141032208238
      "
12845 X1$=X1$+"133003173134003024
      "
12850 X1$=X1$+"105016141134003141
      "
12855 X1$=X1$+"018208144227076049
      234"
12865 X3=0:FOR X1=1 TO LEN(X1$) S
      TEP 3:X3=X3+VAL(MID$(X1$,X1
      ,3)):NEXT:REM  CANCELLARE
12870 IF X3<>8124 THEN PRINT"ERRO
      RE NEI DATA":END :REM  CANC
      ELLARE
12875 X2=0:FOR X1=1 TO LEN(X1$) S
      TEP 3:POKE 828+X2,VAL(MID$(
      X1$,X1,3)):X2=X2+1:NEXT
12885 POKE 889,X0:IF X0$="COLOR"
      THEN 12892
12890 POKE 833,49:POKE 839,234:PO
      KE 849,240:POKE 856,9:POKE
      857,128
12892 SYS828:POKE 53280,14:RETURN
12899 REM  BORDO IN TECNICOLOR
```

12900 Scritta lampeggiante

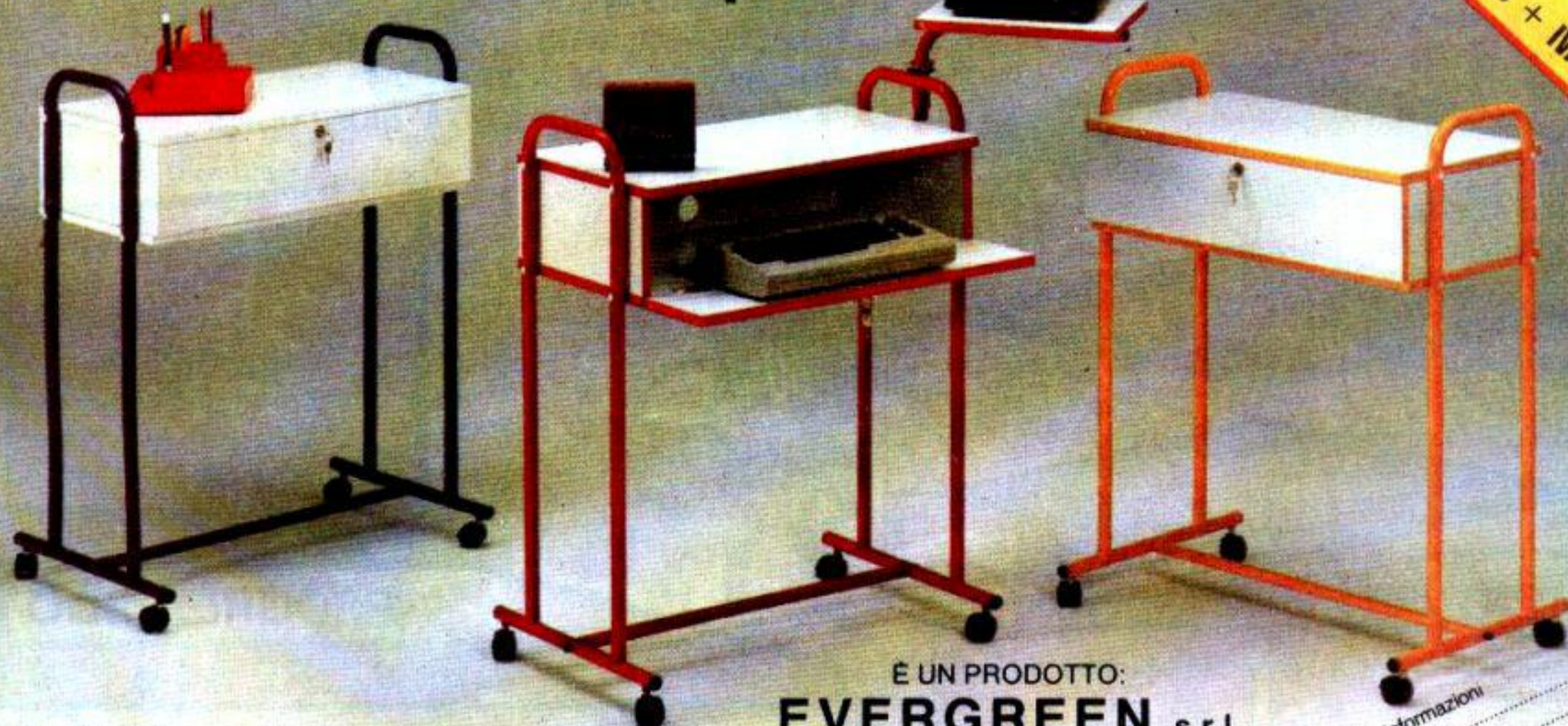
(Per qualsiasi Commodore)

Il nome basta da solo per comprendere lo scopo di questa semplice subroutine, alla quale bisogna inviare i seguenti parametri:

X1$: frase che intendiamo far lampeggiare sul video. Il lampeggio è simulato alternando a X1$ la stringa XS$, contenente spazi vuoti;
X1 e X2: tempo di permanenza e scomparsa del messaggio;
X3: tipo di lampeggio: se X3=1 in rever se, se X3< >1 normale;
X4: distanza del messaggio dal bordo sinistro del video.

Per tornare al programma principale bisogna premere un tasto qualsiasi, che può essere controllato dall'utente tramite XX$. Nel demo d'esempio il programma procede solo se viene premuto il tasto "Z".

Flavio Molinari

```
100 REM  ESEMPIO D'USO
110 REM  SCRITTA LAMPEGGIANTE
120 REM  QUALSIASI COMMODORE
130 :
140 X1$="PREMI 'Z' PER CONTINUA
    RE"
150 X4=10:REM  DISTANZA DAL MAR
    GINE SINISTRO
160 X1=500:X2=300:REM  CICLI DI
     RITARDO PER LAMPEGGIO E PA
    USA
170 X3=1:REM  TIPO LAMPEGGIO: R
    EVERSE
180 GOSUB  12900:IF  XX$<>"Z" T
    HEN 180
190 :
200 :
9999 END
12900 IF XX$<>"" THEN 12910
12902 XS$="":FOR XX=1 TO LEN(X1$)
      :XS$=XS$+CHR$(32):NEXT
12905 IF X3=1 THEN X1$=CHR$(18)+X
      1$+CHR$(146)
12910 PRINTCHR$(145) TAB(X4)XS$:F
      OR XX=1 TO X1:NEXT
12915 GET XX$:IF XX$<>"" THEN RET
      URN
12920 PRINTCHR$(145) TAB(X4)X1$
```

```
12925 FOR XX=1 TO X2:NEXT
12930 GOTO 12910
12932 :
12980 REM  X1$:STRINGA LAMPEGGIAN
      TE
12981 REM  X1,X2: CICLI PAUSA-LAM
      PEGGIO
12982 REM  XX$: TASTO PREMUTO
12985 REM  X3=0: LAMPEGGIO NORMAL
      E
12990 REM  X3=1: LAMPEGGIO IN REV
      ERSE
12999 REM  SCRITTA LAMPEGGIANTE
```

Elenco delle routine pubblicate

Qui di seguito viene riportato l'indice di tutte le routine finora pubblicate. Fra parentesi il numero di Commodore Computer Club su cui sono apparse.

63970 REM 50500 VISUALIZZA FILE (28)
63971 REM 50400 LEGGE FILE RELATIVI (28)
63972 REM 50300 SCRIVE SU FILE RELATIVI (28)
63973 REM 50200 CREA FILE RELATIVI (28)
63974 REM 50000 LEGGE BLOCCHI LIBERI (28)
63975 REM 12200 NUMERI CONGRUI (28)
63976 REM 12100 PROTEZIONE SOFTWARE (28)63977 REM 12000 KOALA (27)
63978 REM 11900 SCAMBIA PAGINA VIDEO (27)
63979 REM 11800 SALVA RAM (27)
63980 REM 11700 CALCOLATRICE (27)
63981 REM 11600 SCOMPOSIZ.SILLABE (27)
63982 REM 11500 CAR.HI-RES (27)
63983 REM 11400 ISTOGRAMMI (27)
63984 REM 50100 ESAME DIRECTORY (26)
63985 REM 11300 FUNZ.INV.IPERBOLICHE (26)
63986 REM 11200 FUNZ.INV. TRIGONOM. (26)
63987 REM 11100 FUNZIONI INVERSE (26)
63988 REM 11000 FUNZIONI IPERBOLICHE (26)
63989 REM 10900 CONVERSIONE DEC-ESA (26)
63990 REM 10800 CONTROLLO DATA (25)
63991 REM 10700 IMPULSI SONORI (25)
63992 REM 10600 REVERSE SCHERMO (25)
63993 REM 10500 INPUT CONTROLLATO (25)
63994 REM 10400 INCOLONNAMENTO VIRGOLA (25)
63995 REM 50000 N. BLOCKS FREE(DISCO) (24)
63996 REM 10300 INPUT & CONTR/DEFAULT (24)
63997 REM 10200 ESTRAZ.PAROLA DA FRASE (24)
63998 REM 10100 CAMBIA COL.BORDO/FONDO (24)
63999 REM 10000 CORNICE POLICROMA (24)

I filtri passivi

di Piero Dell'Orte

Come analizzare la curva di risposta di un filtro passivo.

Chiunque conosca l'elettronica di base può comprendere le formule del listato. Per gli altri, rimane in risalto l'aspetto didattico del programma.

Ecco comunque qualche cenno sui filtri e sul programma. I filtri sono quadripoli che hanno la proprietà di lasciar passare i segnali di determinate frequenze e bloccare quelli delle altre. Vengono qui esaminati due tipi di filtri:il passa-basso il passa-alto. Questi, scelti del tipo a RC (resistenza-condensatore), permettono il passaggio del segnale, inalterato o quasi, per tutte quelle frequenze che siano inferiori o superiori a una frequenza chiamata frequenza di taglio.

Viene definita frequenza di taglio quella frequenza alla quale l'amplificazione si riduce della quantità di radice di due (0.707) rispetto al massimo segnale la cui ampiezza si considera unitaria per convenzione.

La formula della frequenza di taglio è la seguente:
ft=1/6.28*R*C

La formula dell'amplificazione è diversa da filtro a filtro; per il filtro passa-basso è uguale a:
A=1/SQR(1+(6.28*f*R*C)↑2);
mentre per il passa alto è:
A=1/SQR(1+(1/(6.28*f*R*C)↑2));
dove f è la frequenza che varia; R e C sono i valori di resistenza e capacità inseriti all'inizio.

Come gira il programma

Il programma, inserendo i valori di resistenza e capacità, permette di tracciare la curva di risposta del filtro in esame. Affinchè la curva sia davvero realistica, viene fatto un controllo sul valore della frequenza di taglio perchè il programma abbia un valore ragionevole di riferimento su cui disegnare.

Essendo il grafico in alta risoluzione, non si può sapere a priori a quale valore corrisponde il punto disegnato in quel momento. Tuttavia, premendo F1, l'esecuzione del disegno viene sospesa, si torna alla pagina di testo e vengono visualizzati i valori "attuali" della frequenza e del segnale. Premendo di nuovo F1 si torna alla pagina grafica e il disegno riprende il suo tracciamento.

Attenzione

Per far funzionare il programma è necessario caricare dapprima le routine grafiche di TOMA e lanciarle. Tali routine sono state pubblicate sul N.14 di C.C.C. e sulla rivista su nastro Commodore Club N.4

Quindi si carica il programma FILTRI e si dà il RUN.

A questo punto verrà chiesto di scegliere un tipo di filtro. Si dovranno ora inserire i valori di resistenza e capacità, dopodichè non resta che osservare il tracciamento della curva e analizzarla.

```
100 REM  ***************************
110 REM  *     COMMODORE     64     *
120 REM  *                          *
130 REM  *     FILTRI  PASSIVI      *
140 REM  *                          *
150 REM  * DI PIERO DELL'OSTE       *
160 REM  ***************************
170 REM  *   CARICARE LE ROUTINE    *
180 REM  *    GRAFICHE DI TOMA      *

190 ←CLEAR
195 DIM  A(1,20)
200 PRINTCHR$(147):POKE  214,10
210 PRINT  TAB(10)"FILTRI PASSIVI"
220 PRINT:PRINT:PRINT:PRINT
230 PRINT"1) PASSA BASSO":PRINT
240 PRINT:PRINT"2) PASSA ALTO"
250 PRINT:PRINT:INPUT "QUALE";N
260 PRINT CHR$(147);IF  N>2  THEN

     200
270 INPUT "RESISTENZA (OHM)";R
280 INPUT "CONDENSATORE (NANOFARAD
    )";C
290 C=C*1E-9
300 FT=1/(6.28*R*C)
310 IF  N=2  THEN  460
320 REM  PASSA-BASSO
330 IF FT<1000  THEN  PRINT"VALORE
     TROPPO BASSO PER UN FILTRO PA
    SSA-BASSO":GOTO 110

340 IF  FT>1E6  THEN  PRINT"FREQUE
    NZA TROPPO ELEVATA":GOTO  270
350 F=0:P=1:X=-150:Y=40:GOSUB  540
360 A=1/SQR(1+((6.28*F*R*C)↑2))
370 A1=INT(A*100):GOSUB  600
380 IF F>1E1 THEN  P=F/6
390 IF F>1E2 THEN  P=F/12
400 IF F>1E3 THEN  P=F/18
410 IF F>1E4 THEN  P=F/24
420 IF F>1E5 THEN  P=F/30
430 F=F+P:IF  A<.2  THEN  630
440 GOTO  360
450 REM  PASSA-ALTO
460 IF  FT>300  THEN  PRINT"FREQUE
    NZA TROPPO ELEVATA":GOTO  270
470 IF  FT<10  THEN  PRINT"FREQUEN
    ZA TROPPO BASSA":GOTO  270
480 F=1:P=1:X=-150:Y=40:GOSUB  540
490 A=1/SQR(1+(1/((6.28*F*R*C)↑2))
    )
500 A1=INT(A*100):GOSUB  600
510 IF  F>300  THEN  630
520 F=F+P
530 GOTO  490
540 POKE  53280,12
550 ←GRAF 12,0
560 ←COL OR  1
570 ←DRAW -160,-60,0,160,-60,0
580 ←DRAW -150,-80,0,-150,80,0
590 RETURN
595 RETURN
600 GET  A$:IF  A$=CHR$(133)  THEN
     750

605 X%=0
610 ←PLOT X,Y,0
620 X=X+1:Y=-60+A1:RETURN
630 POKE  53280,11
640 GET  A$:IF  A$="" THEN  640
650 ←TEXT 6,14:PRINT CHR$(147)
660 PRINT"ORA PUOI FARE:"
670 PRINT:PRINT"1) DISEGNARE SOPRA
     LA PRECEDENTE"
680 PRINT:PRINT"2) DISEGNARE UNA S
    OLA CURVA "

690 PRINT:PRINT"3) FINIRE"
700 PRINT:INPUT "QUALE";M
710 IF  M=3  THEN  END
720 IF  M=2  THEN  190
730 IF  M=1  THEN  200
740 GOTO  700
750 PRINTCHR$(147):←TEXT 6,14
755 PRINT"RESISTENZA="R
756 PRINT"CAPACITA'="C"NANOFARAD"
760 PRINT:PRINT:PRINT TAB(10)"FREQ
    UENZA DI TAGLIO"

770 PRINT:PRINT TAB(15);INT(FT);"
    HZ"
780 PRINT:PRINT:PRINT"FREQUENZA";I
    NT(F);" HZ"
790 PRINT:PRINT"SEGNALE A";INT(A*1
    00)/100
795 A(0,X2)=F:A(1,X2)=A:X2=X2+1
796 FOR X1=0 TO (X2-1)

797 PRINTA(0,X1);A(1,X1):NEXTX1
800 GET  A$:IF  A$<>CHR$(133) THEN
     800
810 ←DRAW X,-60+A1,0,X,-60,0
820 GOTO  540
```

Vendo C16, giochi, 2 libri a sole lire 170.000. *(Walter Astori, corso Orbassano 88, 10136 Torino. Tel. 011-392413).*

Vendo Commodore Plus 4. *(Andrea Pruneri, via Val di Sole 14 - 20141 Milano. Tel. 02-5392520).*

Vendo C64, 350 programmi a lire 700.000. *(Guido Scalmato, via Trilussa 37, 04011 Aprilia. Tel. 06-923447).*

Vendo stampante Commodore VPS 803, lire 390.000 trattabili. Solo zona Friuli. *(Aristide Vetere Rossi, via V. Veneto 13, 33097 Spilimbergo. Tel. 0427-40315).*

Vendo CBV 64, registratore dedicato, Drive 1541, stampante MPS 803, programmi e libri vari. *(Nicola Lorizzo, via Pisani 42, 70031 Andria).*

Compro espansione 8?16 K per Vic20. *(Alberto Ryolo, via Alciati 7, 20146 Milano. Tel. 02-4150893).*

Vendo Vic20, registratore, introduzione basic, 2 cartucce originali, 2 cassette Mastertronics, programmi vari il tutto a lire 200.000. *(Daniele Romano, via Oreto Nuova 462, 90124 Palermo. Tel. 091-440729).*

Vendo Commodore 64, drive 1541, registratore 1530, 2 joystick autofuoco, il quik disk della Philips. Il tutto a lire 900.000. *(Stefano Russolillo, via Palo Laziale 53, 00055 Ladispoli. tel. 9913445).*

Vendo miglior offerente Commodore 16 (tastiera, registratore), giochi. *(Riccardo De Santis, via G. Pazienza 21, 71016 San Savero - Foggia. tel. 0882-21529).*

Vendo CBM 64, 2 manuali, tasto reset; cassetta con pascal, Forth, L/M, Simon basic, 4 turbo, 10 giochi a scelta. Il tutto per lire 400.000. *(Luca Minudel, via Kennady 11, 31015 Conegliano - Treviso. Tel. 0438-34272).*

Cerco la cartuccia con microprocessore Z-80 da inserire sul Commodore 64, più l'assemblatore Z-80. *(Vincenzo Cupaiolo, piazza Bruno Pompei 1, 00040 Ariccia - Roma. Tel. 06-9!30947).*

Vendo monitor a colori Cabel 14". *(Vittorio Zanoni, via Carlo Cattaneo 42, 22063 Cantù. Tel. 031-711317).*

Vendo Commodore Vic20, joystick, cassette giochi. Il tutto a lire 150.000. *(Raul Bellomi, via Jean Jaures 8, 20125 Milano. Tel. 02-2894197).*

Vendo Commodore Vic20, registratore originale, joystick, padles, varie cassette di programmi. Il tutto in perfette condizioni completo di cavetti a lire 100.000. *(Claudio Bruzzese, via degli Olmi 71, 00172 Roma. Tel. 06-2300405).*

Vendo Commodore Plus/4 completo. *(Giorgio Sabbatini, corso XX Settembre 18, 61043 Cagliari. Tel. 0721-782789).*

Cerco ZX Spectrum 48K con registratore a lire 160.000. Cerco allo stesso prezzo stampante per Vic20, plotter. *(Alessandro Mirri, via Roberto San Severo 7, 00176 Roma. Tel. 06-2753830).*

SCAMBIATEVI LE LISTE

Waltar Ragfaelli - via Mazzini 125 - 26013 Crema.

Mauro Ferri - via Croce Coperta 12 - 48022 Lugo - tel. 0545/27637.

Simone Bordet - viale Biella 27 - 10015 Ivrea - tel. 0125/251572.

Luca Bernardini - viale dello Stadio 21 - 05100 Terni - tel. 0744/422141.

Gianluca Giustiniani - via Raffaele De Cesare 6 - 00179 Roma.

Carlo Pezza - via S. di Santarosa 61 - 00149 Roma - tel. 06/5281016.

Bruno Di Gese - via Celentano 52 - 70121 Bari - tel. 336778.

Andrea Belluzzi - via Voltome 52 - 47031 San Marino - tel. 0541/991953.

Francesco Liperoti - via A. Grandi 22 - 22040 Sirone - tel. 031/850713.

Luca Cappellini - via Marco Polo 7 - 20100 Milano - tel. 02/6571523.

Bruno Mialich - via Cardinal Jacopo Monico 23 - 30174 Mestra (Venezia) - tel. 041/914301.

Ciro Gasparre - via Camaldoli 12/A - 80131 Napoli - tel. 081/463867.

Centro Programmazione Software via Francesco crispi 221 - 91010 Salemi.

Francesco Arcidiacono - via Acquedotto del Peschiera 86 - 00135 Roma - tel. 06/334746.

Massimiliano Moratto - via Carlo Collo 1 - 40133 Bologna - tel. 051/569194.

Gianfranco Deroma - via Italo Simoni 2 - 07100 Sassari - tel. 070/272871.

Claudio Lestan - via Dei Leoni 42/2 - 34170 Gorizia - tel. 0481/32843/20366.

Lucio Rossetto - via delle Alghe 3 - 30126 Lido di Venezia - tel. 765639.

Boss Club - via Perosi 10 - 61032 Fano (Pesaro) - tel. 0721/866940.

Marco Martinelli - via Valtrighe 6 - 24030 Terno d'Isola (BG) - tel. 035/905196.

M. Lusardi - via Luigi Brenti 6 - 40057 Bologna - tel. 760073.

Antonino Baglione - via Castellana 318 - 90135iPalermo.

Ugo Riccelli - via B. Corenzio 23 - 84100 Salerno - tel. 35022.

Giovanni Pugliese - via A. Volta 93 - 74100 Taranto - tel. 099/413769.

Giuseppe Spagnolo - via Crispi 37 - 74028 Sava - tel. 099/6708878.

Stefano Pierangeli - via Panaro 15i-00015 Monterotondo - tel. 9000562.

Diego Stabile - Prolungamento C. Demarco 32 - Napoli - tel. 446818.

Cerco urgentemente espansione per Vic20 selezionabile sino a 16 0 32 K. *(Marcello Fruttero, via Valle Andrea 2, 10020 Mombello-Torino. Tel. 9875127).*

Cerco stampante preferibilmente bidirezionale per CBM. *(Norberto Baroni, via Roma 109, 24030 Vercurago - Bergamo. Tel. 0341-420581).*

Vendo C64, registratore, copritastiera, joystick, 200 programmi a lire 350.000. *(Emidio Borzi, via Cellini 44, 74029 Talsano - Taranto. Tel. 099-511761).*

INVIARE TUTTA LA PAGINA ANCHE SE SI UTILIZZA UNA SOLA SCHEDA

Nome .... Cognome ....

Via .... N° .... CAP. .... Città ....

Telefono .... Orario ....

Registrate il mio abbonamento annuale a Commodore Computer Club.

☐ Ho versato oggi stesso il canone di L. 35.000 a mezzo c/c postale **n°37952207** intestato a:
Systems Editoriale - V.le Famagosta, 75 - 20142 Milano

☐ Ho inviato oggi stesso assegno bancario n. ....
per l'importo di L. 35.000 intestato a Systems Editoriale

Si prega di scrivere il proprio nome e l'indirizzo completo in modo chiaro e leggibile. Inviare la fotocopia del bollettino di c/c postale.

Considerando che i numeri 1, 2 e 7 sono esauriti, vogliate inviarmi i numeri arretrati .... al prezzo di L. 5.000 cadauno per richieste fino a 4 numeri, o di L. 4.000 cadauno per richieste oltre i 4 numeri arretrati, e perciò per un totale di L. .... Sono a conoscenza che i fascioli suddetti non saranno inviati in contrassegno e, pertanto, ho provveduto oggi stesso a versare il canone di L. .... a mezzo c/c postale n. 37952207 intestato a:
Systems Editoriale - V.le Famagosta, 75 - 20142 Milano

STATISTICA

Non posseggo un computer ☐

Posseggo un C64 .... si ☐ ... no ☐
Posseggo un VIC 20 .... si ☐ ... no ☐
Posseggo un Commodore Plus 14 .... si ☐ ... no ☐
Posseggo un Commodore Plus 16 .... si ☐ ... no ☐
Posseggo un registratore dedicato .... si ☐ ... no ☐
Posseggo un drive 1541 .... si ☐ ... no ☐
Posseggo una stampante .... si ☐ ... no ☐
Posseggo un monitor .... si ☐ ... no ☐

COLLABORAZIONE

A titolo di prova vi invio un articolo e la cassetta.........................disco..............................
col programma che intendo proporre per la pubblicazione di cui garantisco l'originalità.

DOMANDA/RISPOSTA ....

....

....

....

....

....

RICHIESTA ARGOMENTI

Mi farebbe piacere che Commodore Computer Club parlasse più spesso dei seguenti argomenti:

1/ ....

2/ ....

3/ ....

4/ ....

GIUDIZIO SUI PROGRAMMI DI QUESTO NUMERO

Ho assegnato un voto da 0 a 10 ai programmi che indico di seguito:

A/ .... Voto ....

B/ .... Voto ....

C/ .... Voto ....

D/ .... Voto ....

PICCOLI ANNUNCI ....

CERCO/OFFRO CONSULENZA ....

INVIARE IN BUSTA CHIUSA E AFFRANCANDO SECONDO LE TARIFFE VIGENTI A:

COMMODORE COMPUTER CLUB

V.le Famagosta, 75
20142 Milano

INVIARE TUTTA LA PAGINA ANCHE SE SI UTILIZZA UNA SOLA SCHEDA

Nome ________ Cognome ________

Via ________ N° ____ CAP. ____ Città ________

Telefono ________ Orario ________

24 ORE SU 24 DI MUSICA IN STEREOFONIA CON

CONCESSIONARIA
PER LA PUBBLICITÀ DI MILANO

RADIANT S.P.A.

CONCESSIONARIA
PER LA PUBBLICITÀ DEL CIRCUITO

gamma color italia S.r.l.

PALAZZO CANOVA CENTRO DIREZIONALE MILANO 2 - 20090 SEGRATE (MI)
TEL. 02/2155714 - 2155726 - 2155734

LOMBARDIA

Milano	95.9-92.8-97.1
Bergamo	99.3
Brescia	92-92.7
Como	97.1
Cremona	99.3
Pavia	95.9-97.1
Varese	94.9

LIGURIA

Genova	96.25
La Spezia	98.7

EMILIA ROMAGNA

Bologna	88.7
Piacenza	97.1

PIEMONTE/VAL D'AOSTA

Alessandria	104.3
Cuneo	90.6-97.6
Novara	97.1
Aosta	91.8-92

TOSCANA

Firenze	97.6-104.4
Livorno	98.2-97.3-100.6
Massa C.	98.7
Pistoia	97.6-104.4
Pisa	97.3
Lucca	97.3

LAZIO

Roma	99.5

systems